Aplicación de HMMs para clasificar series temporales

TFG de Ingeniería Informática

Autor: Faustino Tello Caballo

Director del proyecto: Jesús García Herrero
Co-director del proyecto: Julia Sidorova

20/06/2014

Contenido

1. Resumen en ingles (Abstract)

There are numerous ways of classifying, each has their specific functions and they work better or worse depending on the environment and other factors. The issue we are going to address in this paper is the classification of time series. It is a sequence of observations or chronological ordered data, where it isn´t necessary for all of the data to have the same intervals. There are a range of methods that can be used to classify time series: artificial neural networks, Bayesian networks... In this case we are going to focus in Markov's hidden models. With Markov's hidden models (HMM), we can use different algorithms for the classification but we are going to use more specifically Viterbi's algorithm.

The time series can be almost every chronological ordered data succession, however in this investigation we are going to use real and pretend flight trajectories, and we will classify the distinct events respect of their direction. The conjured data is created by a simulator that is able to generate a flight destiny, incorporate radars and integrate different levels of noise and resolution. In addition how this happens in real life, these levels increase depending on the distance with the radar. The pretend radar captures the data in a period of time and then creates the time series. The real trajectories are from various airports with radars, which collects and storages the data for the subsequent use.

The current investigation has several objectives that we will tackle during this paper, the main one is to probe Markov's (HMM) functioning with different time series and see its efficiency on real data.

- Firstly, with a basic case we will probe the limits of a HMM for different time series, how distinct factors can influence in the series' length and how it affects the classification. We will also keep changing the probability matrix, transitions and the observations, to see how it can affect and modify the model that passes to the algorithm the classification's result.

- Secondly, we are going to use some ideal trajectories created by a simulator to obtain a classified data without mistakes, and following create a confusing matrix and see the percentage of correct answers when classifying other trajectories.

- Thirdly, we will create trajectories with different levels of noise and resolution to probe where the algorithm can obtain a good percentage of right answer. After the data will be passed on a Kalman filter and the process will be process.

- Fourthly and last, we will coach the model to classify the real trajectories before and after the application of Kalman's filter. The problem in this last section is that the real data does not offer an ideal classification, so analyzing the percentage of correct answers is more complicated.

1.1. HMM's Explanation

This models are like others Markov Model's [1], the states are unobservable, but we can see other state called observable states. The observable states have a dependent relationship with the unobservable states, then we can estimate the unobservable state with the other.

- The unobservable states, $S = \{S_1, S_2, S_3 \ldots S_n\}$.
- The observable states, $V=\{V_1,V_2,V_3 \ldots V_m\}$.
- The observation matrix, $B=\{b_j(k)\}$.
- The transition matrix, $A=\{a_{ij}\}$.
- The initial state vector, $\pi = Pr(X_1 = S_{1..n})$.

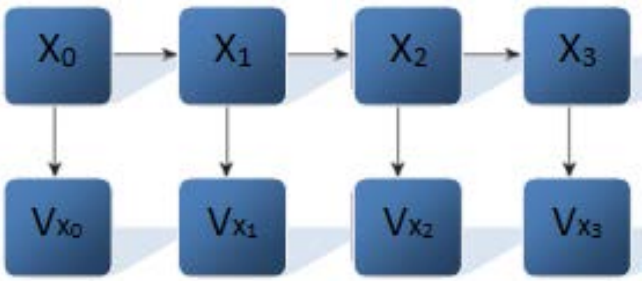


Figure 4:Hidden Markov Model

With the model and the Viterbi algorithm's, we can have the classify of data.

1.2. Problem's context

For the introduction I have used a toolbox called Murphy HMM. This toolbox is compatible with discreet Gaussian exit, or a mixture of Gaussians. In addition it is compatible with the inference and learning. In this case our exits will be discreet, as we want to know the plane's actions in every moment.

We will have different conjured files with different levels of noise and distinct resolution sizes, besides other ones where I have used Kalma's filter to use HMM's velocity. The resolution has the same effect than in the images: it increases as the number of cells does, and it decreases the same way. The same happens with cells, as each cell has a colour, when the number of cells increments there are more tonalities in the image, what allows it to be seen better. So when we have resolution this means that the points we have, have been discretised in the cells. The bigger it is the cell's size, a spot that does not really occupy the same space than other that can occupy it, it is very similar to make an approximation. In many occasions when we make reference to increase the resolution, we are really referring to increment the cell's size. The problem with resolution is that it tends to grow in curves because the plane decreases its speed, and the sensors get points that are closer between them, and this can superimpose them when resolution increases. The noise it will also be a serious problem, as the angular difference between two points can be totally modified by the noise. Kalman's filter will have a very positive effect with high noise levels, but will also have other problems that we will see in the future. To be finished I will use a file with real data to probe the constructed model's efficiency.

1.3. Manual for the application

We will have three execution files depending on the data we want to classify. If we wanted to classify conjured trajectories with different noise or resolution levels, we will use the file ***simuladasHMM.m***. If we want to classify conjured trajectories in which have been previously applied Kalman's filter and in addition they have different noise or resolution levels, we will use the file called ***simuladasKalmanHMM.m***. If we want to classify real trajectories, independently if they have applied Kalman's filter or not we will use the file ***realesHMM.m.***

Once we know this, we are going to divide the explication in 5 parts, all the files have the same structure, even if they are not identical.

Parameters' election

The files have all the same format, they have a parameters' zone where, according to the data you introduce, you will obtain a starting data that you asked for the algorithm to classify them. Depending on this data we will use one model or another, so it cannot classify the same way data with a minimum noise and data with a lot of noise, the same way it happens with Kalman's filter.

- We have got the file ***simuladasHMM.m*** for all the conjured data, with or without noise.

```
r=5; % 1-5 Grado de Resolucion o Ruido
nr=1;%resolucion 1 || ruido 2
track=1;%1,2,3
sensor=1;
```

- We have got the file ***simuladasKalmanHMM.m*** for the conjured data with resolution or noise that have applied Kalaman's filter.

```
r=5; % 1-5 Grado de Resolucion o Ruido
nr=1;%resolucion 1 || ruido 2
track=1;%1,2,3
sensor=1;
```

- We have got the file ***realesHMM.m*** for real data.

```
filtro=0;% sin filtro 0 || con filtro 1
trayectoria=3;%1-4
Nsensor=1;
```

Data explanation

We are going to work with the following conjured trajectories:

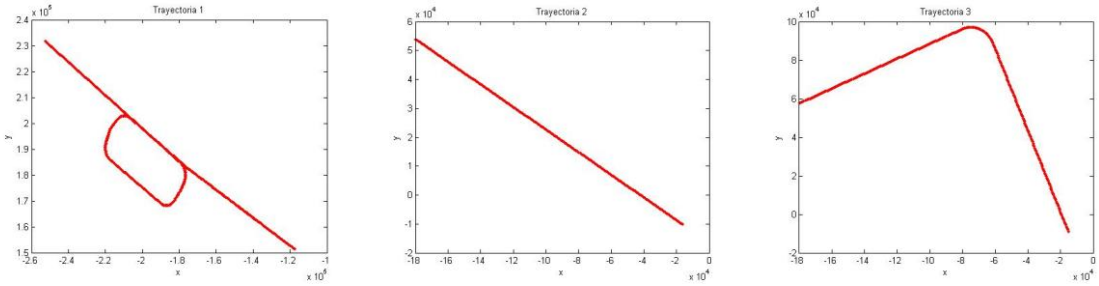


figure 8A, B ,C

As we can see in the first trajectory, it is a hippodrome. It is made by a combination of turns to the right and straights, the trajectory is only one round. The second trajectory is a simple straight line. And third one is a 90 degree turn to the left.. We also have the following real trajectories:

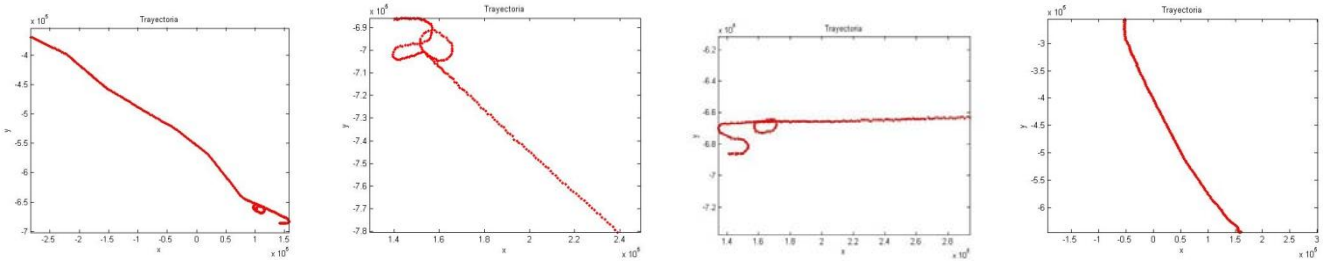


figure 9A, B ,C ,D

Data treatment or observed state creation

For the trajectories that have not been applied Kalman's filter we get positions that are part of the trajectory, and from this ones we will obtain angular differences between the points according to the following formula:

```
for i=1:(length(A)-1),
 x = (A(i)-A(i+1));   %x = A(i)
 y = (B(i)-B(i+1));   %y = B(i)
 angulo =atan2(y,x)*180/pi;
 C(i)= angulo;
 if (i>1)
    dif=C(i-1)-angulo;
    while(abs(dif-360)< abs(dif))
         dif= dif -360;
    end
    while(abs(dif+360)< abs(dif))
         dif= dif +360;
    end
    D(i-1)= dif;
    D(i)=0;
    D(i+1)=0;
 end
end
```

Figure 10: Creation code observable states (without Kalman's filter)

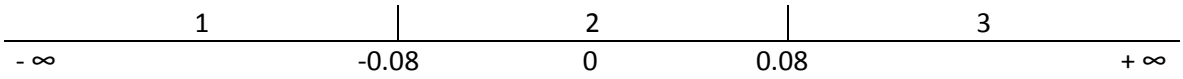
Inside the ringlet that goes over vector A which contains the positions in axis x of the trajectory, vector B has the positions for axis Y. We use the arc tangent and in order to obtain the angle and we pass the radians to degrees. When we get the first angle and from this one we make the difference with the second one, so we can find the angular difference of all the trajectory's positions with the next ones. The vector's D filling that contains the angular differences is produced because it is necessary that is made the same size than A and B if I want to represent it in any graphic (They are not the same size because in order to obtain the angle I need two positions from A, and for the angular difference I need two angles, that is why the length is two values less)

When Kalman's filter is applied and we do not use the position to obtain the angle, we use Kalman's filter to get the angular velocity, and thanks to it we do not have to calculate the angle, and we simply have to make the angular difference. This is caused because Kalman's filter here used the vector velocity to reduce the trajectories' noise. Without this detail the rest is exactly the same.

The result will be continue angular differences, and after it we will discretised to convert them into observations and give them to the algorithm, so this one can estimate the actions in every moment.

The discretization from the angular differences to convert them into observations have three types:

1. Angular difference far from zero and negative.
2. Angular difference close to zero.
3. Angular difference far from zero and positive.



The lines make the discretization edges. The edge between 1 and 2 is -0.08 degrees, while the one between 2 and 3 is 0.08 degrees. The discretization edges will change depending on the data changes to generate the observed state.

Model's election

The model's election is defined by some parameters chosen at the beginning, depending on the file. If they are from resolution or noise, the level of resolution or noise, or if the that has been applied Kalman's filter. Depending on this we will look for a model that previously has been trained with different trajectories with the indicated parameters. The EM algorithm is not used as the obtained results are worse than training manually the model by trial and error.

Algorithm's functioning

The algorithm's entrances are trained HMM and the observed state discretiseds, once it has the Viterbi algorithm data, it classifies the possible data status.

The actions or status that the algorithm will provide as results are the followings:

1. Turn left.
2. Straight.
3. Turn right.

Exits' evaluation

The conjured data when we do not use the ideal data will be evaluated by a confusion matrix, that we will obtain by classifying the ideal trajectories, as in these ones there is any mistake

This way we will get the mistakes and correct answers percentage in each of the possible status, in order to evaluate the result in detail.

1.4. Ideal data

In order to use the toll applied to the problem, we will first use the data obtained by a simulator without noise and resolution. This is caused because the model has to work better when the data does not have noise or any factor that can modify its trajectory. This way we can keep adapting the HMM each time, so the observable states are the most real as possible and can get better in the algorithm's exit.

The data's file includes various trajectories which come from different sensors, in this case the sensor is not important as it will give us the same positions, we will just get a random one for the different trajectories.

For this first case the observation matrix and the transition matrix are the followings:

Next Status / Current Status	Turn left (1)	Straight (2)	Turn right (3)
Turn left (1)	0.65	0.25	0.1
Straight(2)	0.1	0.8	0.1
Turn right (3)	0.1	0.25	0.65

Table 25: Transition matrix of the model with ideal data

Action / Observation	Turn left (1)	Straight (2)	Turn right. (3)
Big negative angular difference (1)	0.55	0.4	0.05
Small angular difference (2)	0.2	0.6	0.2
Big positive angular difference (3)	0.05	0.4	0.55

Table 26: Observation matrix of the model with ideal data

Track 1

We first obtained a graphic with the angular differences. This graphic allows us to see where this differences are, which is very useful where we have to introduce the edges for the discretization. Once we have discretised there are shown in the graphic in the right the angular differences discretised that are observable states when passing them to the algorithm.

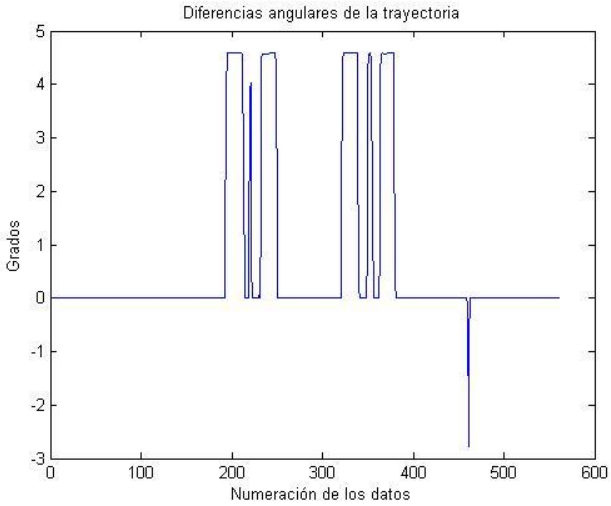


Figure 11A

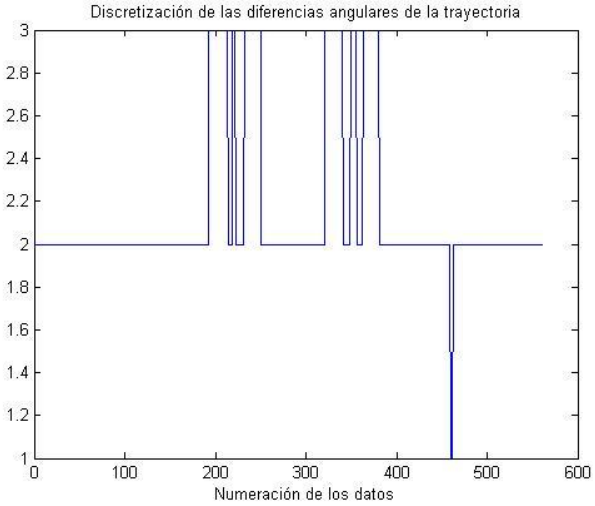


Figure 11B

Once we have the observables and the model ready, we will give them to the algorithm that will give us back the exit, as we can see in the graphic it is a good estimation of the actions done. The trajectory has no turns to the left, so it has not classified any data to 1, while the turn to the right (value 3) have been correctly classified. As we mentioned before the straights are classified as 2.

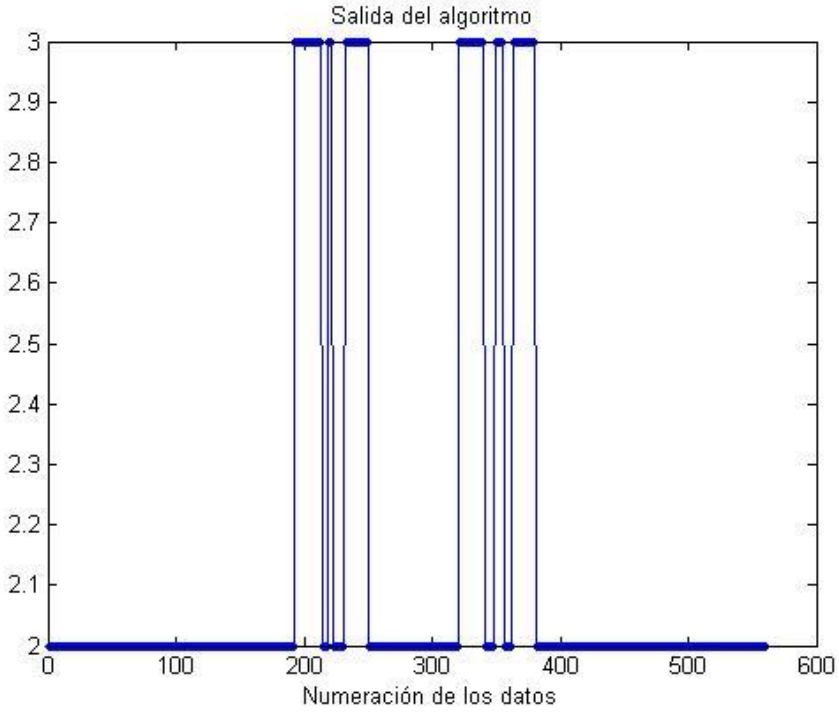


Figura 11C

The next graphic shows the same information but represented in a different way. In addition it shows us the trajectory so we can appreciate the actions classifications that the algorithm gives us. That is, the axis X and axis Y show the trajectory, while Z shows if it is turning to the right (3), straight (2) or turns to the left (1). This way we can easily see if the algorithm classifies good or not, as if we see that the trajectory has a turn and it is in axis Z value 2 it is not well classified.

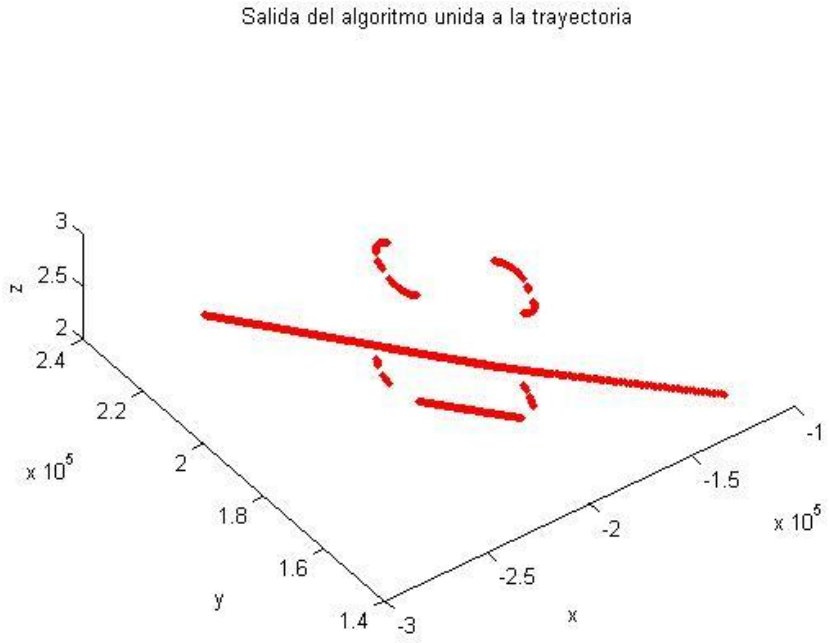


Figura 11D

1.5. Comparison between results and conjured data

Making an average between the three trajectories we obtain the next table with the different parameters that have been used for the simulations.

Types **Percentages**	Resolution	Noise	**Resolution and Kalman's filter**	**Noise and Kalman's filter**
1/8	99.26%	32.26%	94.81%	94.67%
1/4	90.2%	38.1%	93.63%	93.67%
1/2	86.63%	87.8%	92.16%	92.8%
1	73.8%	90.5%	89.83%	90.73%
2	73.2%	90.7%	88.81%	87.67%

Table 87: Percentage of average hits of all simulated trajectories divided by its parameters

Firstly we see how the resolution's increase in the conjured trajectories has an effect that harms the exit in 30% of mistakes in its classification, we see a small step in the averages progression with resolution 1, but apart of this it is normal.

After this, with the algorithm's noise does not obtain any decent result. The high level of right answers has any value in the classification, as it just makes the most numerous action, and as the most numerous action reaches this percentage in approximately 90% of the trajectories. However if the trajectories are modified and we included more turns the percentage would decrease drastically.

With resolution and Kalman's filter applied to the entrance data, we see how at the beginning the classification result is worse, as Kalman's filter it is thinking to soften the jumps in the data generated due to the noise or the resolution. When this jumps are very small the algorithms reacts work better with the original data. Nevertheless we see that the increase in the number of correct answers does not wait when we increase the resolution.

For the noise and with Kalman's filter we see a surprisingly change. From almost not being able to make the classification to make a good one with an 87% of correct answers in the worst of the cases. What it is clear is that the algorithm works much better with the data generated from Kalman's filter, which simply has an elevated noise.

1.6. Real data classification

Trajectory 1

As we do not have any ideal trajectory, we cannot know the exactly percentage of correct answers, but we can graphically see the mistakes and changes. We are going to focus specially in the changes that we have between different trajectories and that sometimes makes the classification better when applying Kalman's filter.

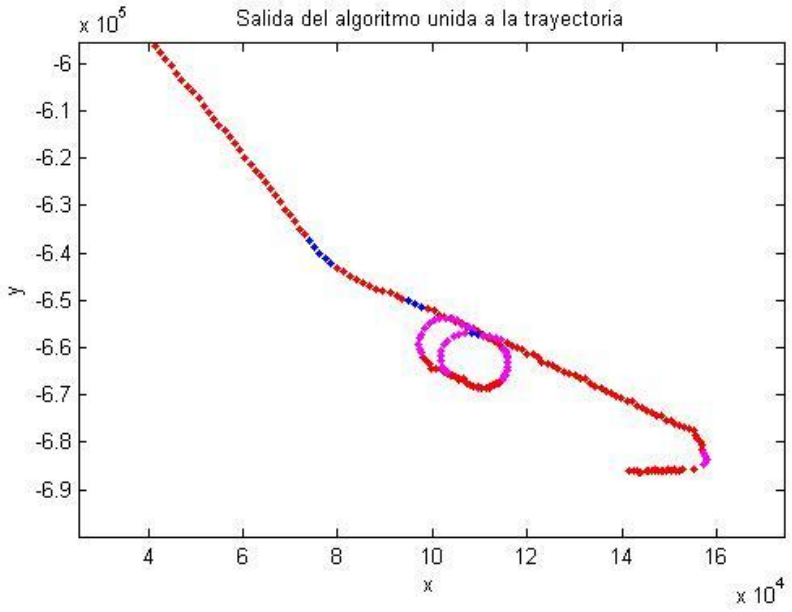


Figure 61: Trajectory 1 without Kalman's filter

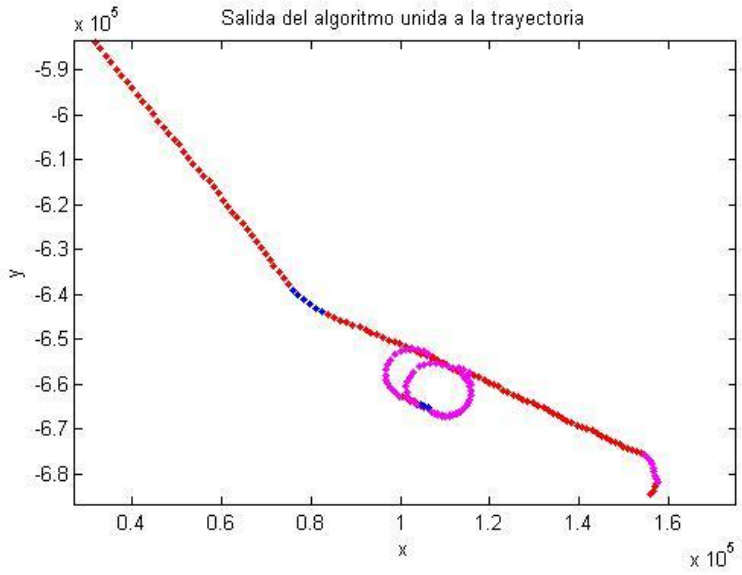


Figure 62: Trajectory 1 with Kalman's filter

As we can see Kalman's filter can modify the trajectory, the same way is does with the noise. However we can suppose that any of the two graphics shows the exactly trajectory because of different factors. We see that the one without Kalman's filter makes a good classification till the last curve, when it does not recognize half of the curve, only a small part, and with Kalman's filter the curve is much more smaller, and it gets a good classification. Nevertheless in the hippodromes classifies the whole hippodrome as a curve, which would not happen as we saw in the first conjured trajectory not all the hippodrome makes the curve. Anyway in trajectory's first part, when the filter is applied, the algorithm distinguishes and classifies some curves that without applying the filter in the entrance data it does not classify correctly. This way, I believe that has a higher number of correct answers in the classification once Kalman's filter is applied in the trajectory.

Trajectory 2

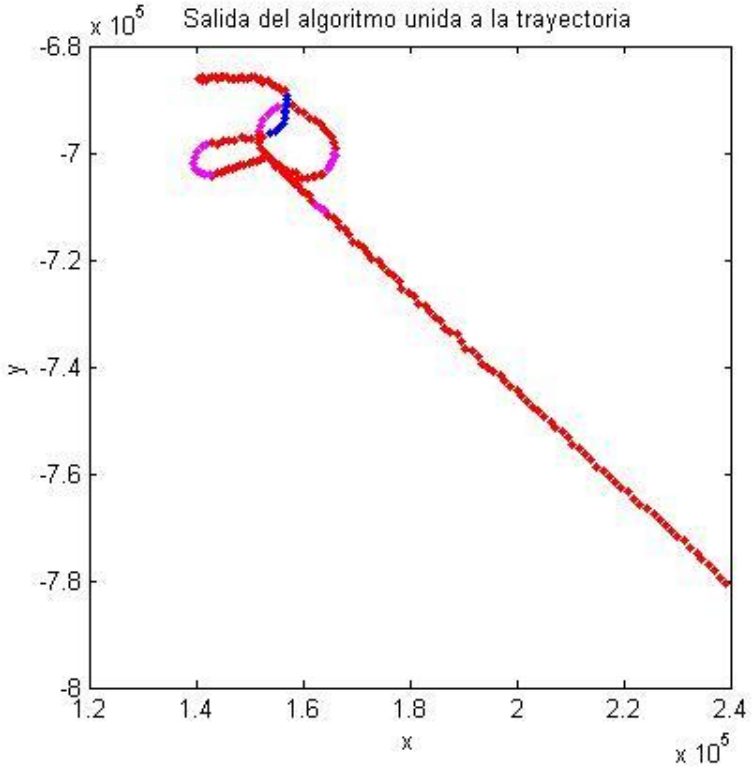


Figure 63: Trajectory 2 without Kalman's filter

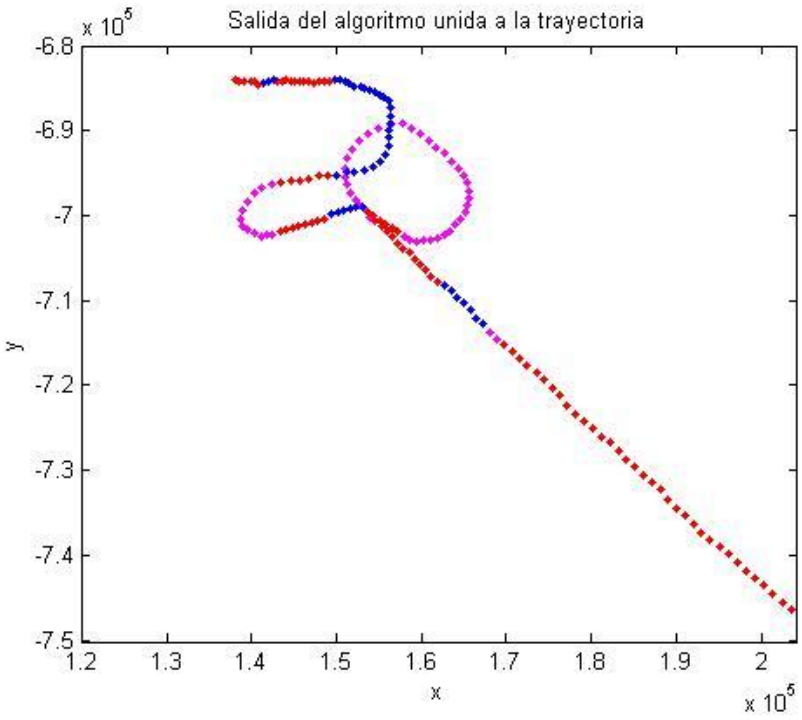


Figure 64: Trajectory 2 with Kalman's filter

We have got the same problem we had with the hippodromes in trajectory 1: without applying the filter the data does not register the whole curve, but once applied the filter it takes the whole hippodrome as a curve. However we can see how when applying Kalman's filter in this case the result is much better, as it can distinguish one more curve, and makes the curves' size bigger so it makes it more real. In both occasions there is a small mistake in the classification when finishing the longest straight, as is classifies a small part as a curve.

Trajectory 3

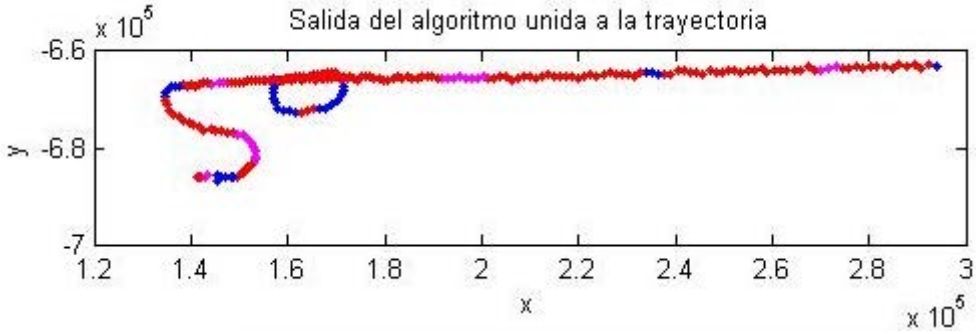


Figure 65: Trajectory 3 without Kalman's filter

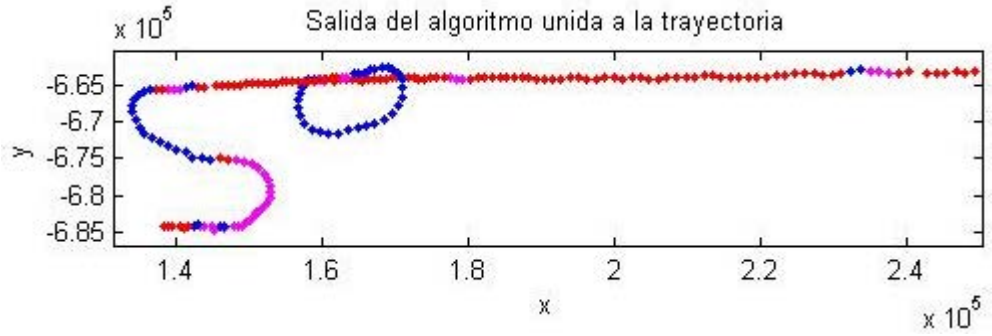


Figure 66: Trajectory 3 with Kalman's filter

We still have the same problems with the hippodromes, but in this case there is a better classification in the hippodromes part without applying the filter. Nevertheless in the following curves this changes, as without applying the filter the classification is not as good as we expected because the noise is very elevated due to the radar's distance. Thanks to the filter the classification gets much more better.

Trajectory 4

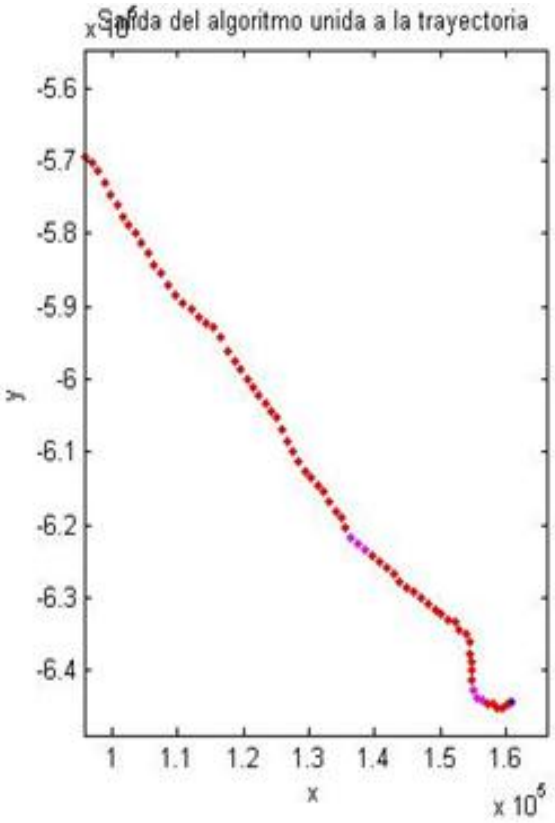


Figure 67: Trajectory 4 without Kalman's filter

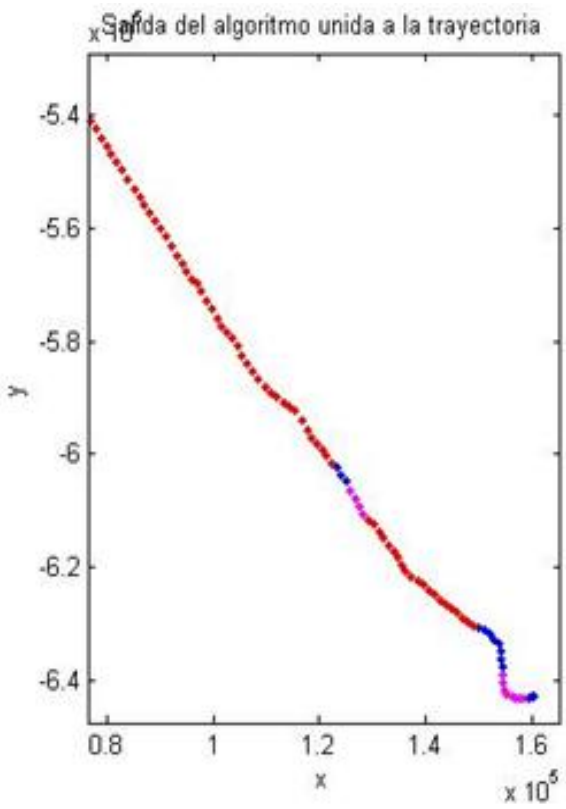


Figure 68: Trajectory 4 with Kalman's filter

This trajectory's classification is much more easier when Kalman's filter is applied, as we need to soften noise's effect because the curves are softer than in the previous trajectories. This way we can affirm it works much more better when there is less noise. If not it would be more difficult to distinguish the noises' curves.

1.7. Conclusions

As we said before the main objective was to probe HMM's efficiency when classifying real planes' trajectories. We have been improving and seeing how we could focus the study and what changes we could introduce to make the efficiency better in the classification. The first problem we had is when we wanted to train the models with the function ***dhmm_es*** as the model was getting overstrained because of the data shortage, and it left very few training circles not reaching very good results. We saw that it was not possible its use after some probes, and by a trial and error method we could manually adjust the models.

After, when we analyzed the models' results with different noise levels, we observed the model's entrance and saw that the problem did not come from the algorithm, it came from the noise, that affected enormously to the angular differences, making impossible a coherent classification. That is why we were looking for a solution and we saw that if Kalman's filter was applied previously to the trajectories, it reduced the noise a lot, even if an inconvenient happened because of the manoeuvre's delay. However it was a good solution to increase the percentage of correct answers.

Nevertheless when we started to work with real data we saw that the noise was not very elevated, and the results were good. But the main problem was the noise difference that we found. The noise increased as the radar was further, and decreased when the radar was closer. This did not allow us to work well the algorithm as it was changing the conditions which the data was recollected, and the algorithm could not recognize the manoeuvres in these zones. By applying Kalman's filter we newly reached a big improvement, anyway in the areas where the noise is higher, when the manoeuvre is straight (it is when the noise effect affects the most) the algorithm confuses sometimes and classifies it as a turn to the left or to the right.

I believe that the results could be improved if we found any more effective training algorithm, able to train the models without the need of excessive data quantity.

2. Introducción

Hay numerosos clasificadores, cada uno tiene sus funciones específicas, actuando mejor o peor dependiendo del entorno y otros muchos factores. El problema que queremos abordar en este documento, es la clasificación de series temporales [2], las cuales son una secuencia de observaciones o datos ordenados cronológicamente, no siendo necesario que todos los datos tengan los mismos intervalos de tiempo. Hay varias métodos para clasificar series temporales, podemos usar redes de neuronas artificiales, redes bayesianas… En este caso, nos centraremos en los modelos ocultos de Markov [1]. Con los modelos ocultos de Markov (HMM), podemos utilizar distintos algoritmos para la clasificación, pero en este caso usaremos el algoritmo Viterbi [3].

Las series temporales pueden ser prácticamente cualquier sucesión de datos cronológicamente ordenados, pero concretamente usaremos trayectorias reales y simuladas de vuelo, y clasificaremos los distintos eventos respecto a su rumbo. Los datos simulados se realizar a través de un simulador por el cual podremos crear una trayectoria de vuelo, introducir unos radares y añadir distintos grados de ruido y de resolución. Además, como ocurre en la realidad, éstos aumentarán en función de la distancia al radar. Los radares simulados captarán los datos a lo largo del tiempo, creando las series temporales. Las trayectorias reales son de varios aeropuertos con radares, los cuales además de utilizarse para controlar el tráfico aéreo, recogen los datos y almacenan para su uso posterior

3. Objetivos

Este proyecto tiene varios objetivos que iremos abordando a lo largo del documento. El propósito principal es probar el funcionamiento de los modelos ocultos de Markov (HMM) con distintas series temporales y ver su eficacia para probar al final con datos reales.

- Primeramente, usando un caso base, probaremos los límites de un HMM para distintas series temporales, y cómo los distintos factores (como puede ser la longitud de la serie) pueden afectar a la clasificación. Iremos cambiando también la matriz de probabilidad de transiciones y matriz de probabilidad de observaciones, para ver cómo puede afectar a los resultados el hecho de modificar el modelo que se le pasa al algoritmo.

- Segundo, vamos a usar unas trayectorias ideales creadas en un simulador para poder obtener unos datos clasificados sin errores, y así poder construir una matriz de confusión posteriormente, y ver el porcentaje de aciertos al clasificar otras trayectorias.

- Tercero, crearemos unas trayectorias con distintos grados de ruido y resolución de los sensores para probar hasta dónde el algoritmo puede obtener un buen porcentaje de aciertos. Y tras esto, pasaremos los datos por un filtro de Kalman y repetiremos el proceso con objeto de tener una estimación de rumbo.

- Cuarto y último, entrenaremos el modelo para clasificar las trayectorias reales antes y después de aplicar el filtro de Kalman. El problema en este último apartado radica en el hecho de tratarse de datos reales, ya que no tenemos la clasificación ideal, por lo que analizar el porcentaje de aciertos de manera exacta es imposible.

4. Estado del arte

4.1. Series Temporales

Como ya hemos mencionado anteriormente, una serie temporal [2] es una sucesión de datos o variables ordenados cronológicamente obtenidos en instantes de tiempo que generalmente están espaciados uniformemente entre sí. El objetivo de analizar las series temporales es obtener patrones de comportamiento, por los cuales se puede predecir la evolución. Si una serie temporal fuera determinista se podría predecir totalmente. No obstante, la mayoría de las series tienen factores aleatorios que sólo permite acercarse a la estructura de la secuencia. Normalmente, las series no son deterministas, o también llamadas estocásticas por distintos motivos, como puede ser la sensibilidad al ruido, la inestabilidad del sistema o la imposibilidad de introducir todas las variables que afectan al sistema en el análisis.

Para realizar el análisis de las series temporales hay varios métodos, pero el más común se basa en la suposición de que el valor de la variable de observación es el resultado de la interacción de cuatro componentes:

- Tendencia secular: es la componente que refleja la evolución de la serie a largo plazo. Generalmente, suele estar en función del tiempo como una función polinómica o logarítmica

- Variación estacional: es la componente que muestra las variaciones u oscilaciones que se repiten en cortos periodos de tiempo de manera periódica y constante.

- Variación cíclica: esta componente recoge las variaciones que se repiten en largos periodos de tiempo. Son variaciones irregulares en función con la tendencia, y son más complicadas de identificar a lo largo del tiempo ya que al ser ciclos más largos son más inestables y se pueden cubrir por distorsionar o cubrir por otros factores.

- Variación aleatoria, ruido o residuo: esta componente está formada por todos los fenómenos accidentales que no muestran ningún patrón regular. Se pueden identificar pero debido a su aleatoriedad es imposible predecirlas excepto regularidades estadísticas (en el ejemplo de las trayectorias no podemos predecir el efecto del ruido, no obstante sabemos que va a ser mayor cuanto más alejado este del radar).

Las series temporales se pueden clasificar en distintos tipos en función de cómo se combinan estos componentes:

- Aditivas: la serie se compone con la suma de los componentes.
- Multiplicativas: la serie se compone con la multiplicación de los componentes.
- Mixtas: la serie se compone por la suma y multiplicación de alguno de los componentes. La más habitual es: $X_t = T_t \cdot E_t \cdot C_t + R_t$. aunque depende totalmente de la serie.

4.2. ATC: Reconstrucción de trayectorias para su evaluación automática

Esto es el dominio del problema [3] [13]. Se basa en una plataforma creada por el grupo GIAA, para un proyecto [11] con el instituto de regulación europeo de tráfico aéreo Eurocontrol, el cual se centra en las técnicas de reconstrucción de la trayectoria para la evaluación de los sistemas ATC (Air traffic control), utilizando datos reales de tráfico aéreo.

Es una plataforma offline que trabaja sobre las grabaciones de datos para análisis y evaluación de prestaciones, a diferencia de lo que se encuentran los controladores (los cuales sólo conocen el estado actual), con esta nueva plataforma tenemos todos los datos de las distintas trayectorias, lo que nos permite hacer segmentación, reconocimiento de patrones, clasificación y muchas otras funciones. Por el contrario, el tener únicamente el estado actual, no nos permitiría realizar estas funciones.

4.3. Clasificadores

Vamos a basarnos en los modelos probabilísticos con tiempo. La principal característica de estos modelos es que el tiempo es discreto, es decir, usando sucesiones de instantes de tiempo. La variable usada se muestra siempre en función del tiempo X_t, ya que esta se estudia a lo largo del tiempo.

Aproximaciones previas

Este problema ha sido abordado previamente con otro tipo de clasificadores, se usaba un sistema de reglas [12] para segmentar las trayectorias. La base de esta segmentación es poder distinguir las distintas maniobras realizadas por la aeronave.

Redes de neuronas

Como ya sabemos, las redes de neuronas [6] son un método muy recomendado para clasificar datos. Su diseño es muy sencillo, y pueden utilizarse también como regresión. Su estructura es simple si hablamos del modelo Adaline, pero si buscamos algo más complejo que no esté acotado por una función lineal, necesitamos un perceptrón multicapa, y la estructura de este se complica. No obstante, las redes de neuronas tienen un problema: para funciones complejas necesitan una base de datos extensa para conseguir buenos resultados, y opuestamente, si es muy pequeña se produce sobre-entrenamiento antes de conseguir la función de salida deseada mediante el ajuste de los parámetros.

Redes bayesianas

Con las redes bayesianas [5] se supone que el estado de la variable X_t depende de X_{t-k} , es decir, de las *k* variables posteriores al instante t . Esto provoca que el número de parámetros crezca exponencialmente en k.

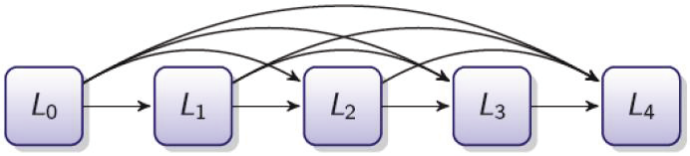


Figura 1: Modelo de Redes Bayesianas

Las redes bayesianas, cuando k es muy elevado se vuelve ineficaz, y la solución a este problema es la suposición de Markov.

Suposición de Markov

La suposición de Markov establece que el estado depende únicamente del estado anterior, no de todos los estados anteriores. Es decir, lo que expresa la fórmula es que la probabilidad de L_t depende solamente de L_{t-1}.

Figura 2: Suposición de Markov

$$P(L_t \mid L_{0:t-1}) = P(L_t \mid L_{t-1})$$

Si añadimos que la información la obtenemos de un sensor, la fórmula correspondiente a la suposición de Markov es la siguiente, ya que el estado depende de la información que tenemos de éste y del estado anterior.

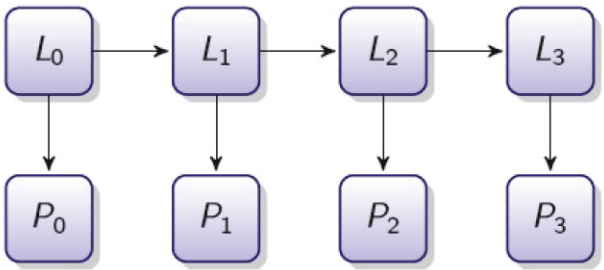


Figura 3: Suposición de Markov con sensores

$$P(P_t|L_{0:t}, P_{0:t-1}) = P(P_t|L_t)$$

Trabajando con la suposición de Markov existen diversos modelos: están los modelos de Markov, los modelos Markovianos de decisión (MDP), una variación de estos son los POMDP, los cuales trabajan con observaciones parciales de los estados. Cada uno tiene sus características, por ejemplo estos dos últimos son modelos de decisión, no se usan como clasificadores, sino que buscan decidir la acción con mayor probabilidad dados unos factores. En los modelos de Markov se conoce el estado actual por lo que tampoco podemos trabajar con ellos. Nosotros nos centraremos en los modelos que sean estacionarios y que los estados sean inobservables.

4.4. Modelos temporales o estacionales

Son modelos en los que no importa el instante de tiempo exacto, en estos modelos se mantiene la misma distribución de probabilidades. Para describir el modelo se necesita una probabilidad inicial $P(L_0)$ la probabilidad de transición de un estado a otro $P(L_t|L_{t-1})$ y la probabilidad de que la observación del sensor influya en el estado $P(P_t|L_t)$.

Sabiendo esto, tenemos varios modelos temporales que además cumplen la suposición de Markov, el primero son los Modelos ocultos de Markov, también tenemos el filtro de Kalman y por último las Cadenas de Markov.

Modelos ocultos de Markov (Hidden Markov model, HMM)

Como ya hemos mencionado, son una pequeña variación de los modelos de Markov [1] en la que los estados no se pueden ver, son denominados estados ocultos, pero si se puede ver otra variable llamada observación que tiene alguna relación dependiente del estado oculto. Esto lo convierte en un modelo estocástico doble.

Históricamente, las primeras aplicaciones de los HMMs se hicieron en el campo del procesamiento del lenguaje natural (reconocimiento del habla). Desde entonces los HMMs han alcanzado un papel primordial en todos los campos del saber: economía, biología computacional, tecnologías para el análisis de la voz, historia, etc... [14], con un impacto mucho mayor que el que Markov hubiera imaginado.

Debe resaltarse que los métodos matemáticos de análisis del lenguaje tienen un impacto potencial que va más allá del análisis del habla. Por ejemplo, el análisis sintáctico y su reconocimiento ("grammar inference") se han utilizado recientemente en el diseño de fármacos y de principios químicos activos ("chemoinformatics") [15]-[16].

En este trabajo hemos aplicado los HMMs al reconocimiento de trayectorias como alternativa a los métodos basados en la aplicación de reglas. De esta manera hemos mejorado la fiabilidad de las predicciones y disminuido la complejidad de Kolmogorov (menor tamaño en la descripción del problema), para de esta forma llegar a una formulación estadística del problema.

Vamos a explicar detalladamente todos los componentes de un HMM:

- Denotamos como los posibles estados siendo n el numero de estos, $S = \{S_1, S_2, S_3 \ldots S_n\}$, y al estado oculto X_t.
- Para representar las posibles observaciones siendo m el numero de estos $V=\{V_1,V_2,V_3 \ldots V_m\}$.
- La matriz de probabilidades de transición, $A=\{a_{ij}\}$. Indica la probabilidad que existe de pasar de un $X_{t-1} \rightarrow X_t$, para cada posibles valores de X.
- La matriz de probabilidades de observación, $B=\{b_j(k)\}$. Indica la probabilidad dada una determinada observación de que el estado oculto tenga un valor determinado.
- Vector inicial de estados $\pi = Pr\ (X_1 = S_{1..n})$. Indica una probabilidad de que el primer estado oculto de la serie tenga un valor u otro.

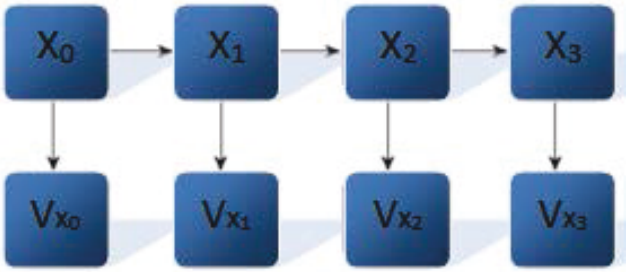


Figura 4: Modelo oculto de Markov

Para crear el modelo necesitamos estos datos, generalmente los valores de los estados y las observaciones vienen definidos por el problema. No obstante, es posible que se necesite hacer un pre-procesamiento de los datos de entrada, y también de los datos de salida para su posterior interpretación.

La matriz de observación, matriz de transición, y el vector inicial se deben crear con las probabilidades, y posteriormente se ajustarán los parámetros, proceso denominado como "entrenamiento del modelo". Existen algoritmos específicos para esto, como son el algoritmo EM y el Baum-Welch, pero se pueden entrenar de muchas formas (no nos vamos a centrar en su explicación ya que no han sido usados en el desarrollo del proyecto). Cabe destacar que funciona en entornos tanto discretos como continuos. Y después de esto, el modelo está listo para su uso.

Entonces usaremos el modelo y se lo pasaremos al algoritmo junto con los datos de entrada para que realice la clasificación de los datos. El algoritmo que usaremos será el algoritmo *Viterbi*.

Algoritmo Viterbi

La función del algoritmo Viterbi [3] es encontrar la secuencia de estados ocultos más probable en un HMM, dado una secuencia de observaciones, es decir, obtiene los estados ocultos que mejor explican las observaciones. Este algoritmo se divide en 4 fases:

Initialization $(i = 0)$: $v_0(0) = 1; v_k(0) = 0$ for $k > 0$.

Recursion $(i = 1, ..., L)$: $v_l(i) = e_l(x_i) max_k(v_k(i-1)a_{kl})$;
$ptr_i(l) = argmax_k(v_k(i-1)a_{kl})$

Termination $P(x, \pi^*) = max_k(v_k(L)a_{k0})$;
$\pi_L^* = argmax_k(v_k(L)a_{k0})$.

Traceback $(i = L...1)$: $\pi_{i-1}^* = ptr_i(\pi_i^*)$.

Filtro de Kalman

Por lo general, es usado para abordar todo tipo de problemas con series temporales. Se puede usar para modelar componentes no observables y parámetros que cambian con el tiempo; pero principalmente se usa para aproximar por máxima verosimilitud.

Se usa en modelos donde los errores se asumen en la medición de los datos. En definitiva, el filtro de Kalman [7] es un procedimiento de predicción y corrección, se encarga de estimar un estado a partir de una estimación previa por el método de mínimos cuadrados añadiendo un valor de parámetro de corrección proporcional al error de predicción, lo que minimiza este error. Se trata de un algoritmo recursivo que maximiza una función de verosimilitud.

Cadenas de Markov

En el caso de las cadenas de Markov [8] es muy similar que los HMM, una vez entendidos. Es el mismo modelo, pero sin observables, es decir, no hay estados ocultos, ni observables que se relaciones con estos probabilísticamente. Esto simplifica el modelo, ya que sólo tendría una matriz de transiciones, los posibles estados y un vector inicial de probabilidades.

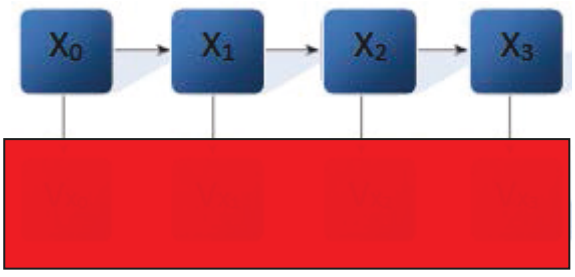


Figura 5: Cadena de Markov

Como podemos ver, se trata de una simplificación de los HMM. En este caso no podríamos usar cadenas de Markov, ya que tenemos observables que se usarán para la clasificación de los estados. No podría depender únicamente del estado anterior.

5. Herramientas basada en los modelos ocultos de Markov.

Para la implementación, he utilizado un toolbox llamado Murphy HMM [8]. Este toolbox es compatible con salidas discretas gausianas o una mezcla de gausianas. Además, es compatible con inferencia y aprendizaje. En este caso, nuestras salidas serán discretas ya que queremos saber las acciones del avión en cada momento.

Las principales funciones de esta librería son las siguientes:

Se utilizará el prefijo dhmm para modelos discretos mientras que mhmm se usa para modelos continuos. El formato será ["salida del la función"] = "función" ("entrada").

1. ***[ll_trace, prior, transmat, obsmat, iterNr] = dhmm_em (data, prior0, transmat0, obsmat0)***: Sirve para el entrenamiento de modelos de Markov discretos. Consiste en la estimación de los parámetros ML/MAP utilizando el algoritmo de EM. El algoritmo de EM, calcula la verosimilitud, y después calcula los estimadores de la máxima verosimilitud de forma que los parámetros maximicen la probabilidad del paso anterior, por lo que podremos trazar una curva de aprendizaje. A esta función debemos pasarle un modelo oculto de Markov.

   El objetivo de esto es una mejora en la capacidad para reconocer las secuencias. Al ser un algoritmo de búsqueda local esto lo hace vulnerable a los máximos locales, por lo que es dependiente de los valores. No se utiliza ya que necesitaríamos una cantidad mucho mayor de los datos, porque con la actual sobreentrena el modelo empeorando los resultados.

2. ***[ll_trace, prior, transmat, mu, sigma, mixmat] = mhmm_em (data, prior0, transmat0, mu0, sigma0, mixmat0)***: Sirve para el entrenamiento de los modelos de Markov continuos.

3. ***[loglik, errors] = dhmm_logprob (data, prior, transmat, obsmat)***: Sirve para clasificar una secuencia dado un modelo discreto, devolviendo la probabilidad logarítmica de que en ese modelo se dé un determinado conjunto de observaciones que previamente se le ha pasado por parámetro, también si en la observación hay estados imposibles indica el error. Es muy útil como método de evaluación de los HMM.

   Como parámetros, se le pasa una secuencia de observaciones, y matrices de probabilidades iníciales, de transición y de observación. Y devuelve la probabilidad de que la secuencia haya sido generada por el HMM.

4. ***[loglik, errors] = mhmm_logprob (data, prior, transmat, mu, sigma, mixmat)***: Sirve para clasificar secuencias dado un modelo continuo.

5. ***[obs, hidden] = dhmm_sample (initial_prob, transmat, obsmat, numex, len)***: Sirve para crear una o más secuencias de observaciones junto con los valores ocultos de los estados para modelos discretos. Como parámetros se le pasan las matrices que definen el HMM discreto junto con la longitud de la secuencia y el número de secuencias que queremos generar.

6. ***[obs, hidden] = mhmm_sample (T, numex, initial_prob, transmat, mu, Sigma, mixmat)***: Sirve para crear una o más secuencias de observaciones junto con los valores ocultos de los estados para modelos continuos.

7. ***[loglik, hidden] = hmmViterbi (data, transmat, obsmat)***: Sirve para determinar los estados ocultos más probables a partir de las observaciones, y además indica la probabilidad logarítmica de toda la secuencia. Este algoritmo será parte importante de nuestro estudio.

En nuestro caso nos centraremos en ***dhmm_logprob*** y ***hmmViterbi***.

- ***dhmm_logprob***: Usará el algoritmo forward-backward. La entrada está formada por las probabilidades iniciales, la secuencia de observaciones y la matriz de transición y de observación. La salida es la probabilidad logarítmica de un conjunto de datos.

- ***hmmViterbi***: La entrada está formada por las probabilidades iniciales, la secuencia de observaciones y la matriz de transición y de observación. La salida será la secuencia más probable de estados ocultos dado los observables y el HMM.

6. Caso de base

Utilizamos un ejemplo académico común para explicar el funcionamiento del toolbox y hemos hecho pruebas con este, con el fin de confirmar su fiabilidad, robustez y sensibilidad con respecto a los parámetros y a los cambios de estos. Esto nos proporcionará unas bases sólidas para comenzar con el uso de la herramienta en nuestro problema.

El problema se llama ***hmm_tiempo***:

Vamos a ir haciendo variaciones del problema para poder apreciar los matices y los cambios que tiene la librería de HMM. En este problema tenemos tres posibles estados que indican el tiempo en una ciudad: que el día esté soleado, nublado o lluvioso.
Estos estados no son observables, pero sabemos que dependiendo del tiempo, una persona realiza unas acciones u otras (pasear, comprar o limpiar) y esta información si la tenemos, por lo que tenemos que hacer es estimar el tiempo en la ciudad dependiendo de las acciones de la persona.

La probabilidad de que pasee si llueve es del 10%, si está nublado del 10% y si está soleado del 60%.
La probabilidad de que compre si llueve es del 30%, si está nublado del 60% y si está soleado del 30%.
La probabilidad de que limpie si llueve es del 60%, si está nublado del 30% y si está soleado del 10%.

Esas serán nuestras probabilidades de observación. Inicialmente está soleado por lo que la matriz inicial es [0,0,1]. También tenemos las probabilidades de transición que son las siguientes.

Si el día anterior ha llovido y la probabilidad de que llueva hoy es del 40% de que esté nublado del 30% y de que haga sol del 30%.
Si el día anterior ha estado nublado la probabilidad de que llueva es del 20% de que esté nublado del 60% y de que haga sol del 20%.
Si el día anterior ha estado soleado la probabilidad de que llueva es del 10% de que esté nublado del 10% y de que haga sol del 80%.

Por tanto tenemos los siguientes parámetros:

```
inicial = [0.0, 0.0, 1]; ESTADOS OCULTOS
transiciones = [0.4, 0.3, 0.3; 0.2, 0.6, 0.2; 0.1, 0.1, 0.8];
probabilidad_observacion = [0.1, 0.1, 0.6; 0.3, 0.6, 0.3; 0.6, 0.3,
0.1];
```

T+1 T	LLUVIA (1)	NUBES (2)	SOL (3)
LLUVIA (1)	40%	30%	30%
NUBES (2)	20%	60%	20%
SOL (3)	10%	10%	80%

Tabla 1: Matriz de probabilidades de transición del caso base

Como podemos observar, nos muestra la probabilidad de que el tiempo cambie o se mantenga según el día anterior, por ejemplo si t = sol tenemos la probabilidad de que t+1= sol de un 80%.

Acciones Tiempo	ANDAR (1)	COMPRAR (2)	LIMPIAR (3)
LLUVIA (1)	10%	30%	60%
NUBES (2)	10%	60%	30%
SOL (3)	60%	30%	10%

Tabla 2: Matriz de probabilidades de observación del caso base

Esta tabla muestra la probabilidad según los estados observables, es decir, las acciones de que el tiempo sea uno u otro. Por ejemplo, si está andando la probabilidad de lluvia es de 10%.

Utilizaremos distintas observaciones, un ejemplo es:

```
observacion2 = [1, 1, 2, 3]; 4 días: anda, anda, compra, limpia
```

También podemos obtener una observación aleatoria generada por el HMM mediante la función dhmm_sample.

```
[obs2, hidden2] = dhmm_sample(inicial, transiciones,
probabilidad_observacion, 1, 5);
```

Esto nos proporcionaría una observación de los estados visibles compuesta por cinco estados ocultos y cinco observaciones que se corresponden.

Para nuestra primera prueba, vamos a utilizar una secuencia de observaciones larga, que nos permita ver las relaciones entre los estados observables y los ocultos. Para esto modificaremos la matriz de probabilidades de las observaciones. Usaremos un 100% de probabilidad para ver la correspondencia.

Acciones / Tiempo	ANDAR (1)	COMPRAR (2)	LIMPIAR (3)
LLUVIA (1)	0%	0%	100%
NUBES (2)	0%	100%	0%
SOL (3)	100%	0%	0%

Tabla 3: Matriz de probabilidades de observación del caso base determinista

Realmente la función Viterbi de la librería no permite un 100% pero usaremos un 99.9%, lo que a efectos prácticos demuestra el funcionamiento de igual manera. Esto sucede porque al ser estados ocultos siempre tiene que haber una grado de incertidumbre.

Usaremos la siguientes observaciones:

				3	1			limpiar, andar
			1	1	2	3		andar, andar, comprar, limpiar
		1	1	2	2	3	3	andar, andar, comprar, comprar, limpiar, limpiar
1	1	3	3	3	2	2	1	andar, andar, limpiar, limpiar, limpiar, comprar, comprar, andar

Tabla 4: Observaciones prueba 1 del caso base

Así cuando estimemos los estados ocultos sabremos perfectamente qué deben ser. Los resultados del algoritmo Viterbi nos ofrece el *path* estimado de estados ocultos y la probabilidad de estos.

				3	1			Path	1 3
								Probabilidad logarítmica	-2.1223
			1	1	2	3		Path	3 3 2 1
								Probabilidad logarítmica	-5.3431
		1	1	2	2	3	3	Path	3 3 2 2 1 1
								Probabilidad logarítmica	-6.7723
1	1	3	3	3	2	2	1	Path	3 3 1 1 1 2 2 3
								Probabilidad logarítmica	-8.8945

Tabla 5: Resultados prueba 1 del caso base

Es visible la relación entre las observaciones y los estados ocultos ya que siempre que se anda (1) también hace sol (3) y cuando se compra (2) también está nublado (2), y además cuando se limpia (3) también llueve (1). Las probabilidades obtenidas también influyen en las transiciones entre los estados ocultos, que veremos más adelante.

Para este otro ejemplo, modificaremos la matriz de transiciones. En este caso, sí nos permite valores de 100%, ya que es posible que algunos estados lleven inevitablemente a otros.

T+1 T	LLUVIA (1)	NUBES (2)	SOL (3)
LLUVIA (1)	40%	30%	30%
NUBES (2)	20%	60%	20%
SOL (3)	0%	0%	100%

Tabla 6: Matriz de probabilidades de transición del caso base (estado oculto 100%)

Con estos cambios obligamos a que cuando un día haga sol el resto de días siga haciendo sol. Así podremos ver que por mucho que cambien los estados observables no cambiarán los ocultos.

Modificamos el estado inicial para que no empiece con sol, y creamos una observación para que hasta que no se ande(1) no esté el estado oculto de sol.

```
inicial = [1, 0.0, 0.0]; ESTADOS OCULTOS
```

Vamos a representar las observaciones usadas junto con la salida del Viterbi y su probabilidad logarítmica:

3 1	Path	1 3
	Probabilidad logarítmica	-2.1223
1 1 2 2 3 3	Path	3 3 3 3
	Probabilidad logarítmica	-15.0215
1 1 2 2 3 3	Path	2 2 2 2 1 1
	Probabilidad logarítmica	-19.0817
3 3 2 1 2 3 2 1	Path	1 1 2 2 2 1 2 3
	Probabilidad logarítmica	-15.3958

Tabla 7: Observaciones y salida de prueba 2 del caso base

Vamos a analizar cada una de las secuencias:

La primera y la segunda son normales, sabemos que cuando está andando hay sol, y cuando hay sol, la probabilidad de que siga haciendo sol es del 100%.

En la segunda, la confianza en esta secuencia es muy baja, ya que teníamos probabilidades de 99.9% de que con observaciones de 2 y 3 surgieran estados 2 y 1, pero la matriz de transiciones lo impide. Por lo que la probabilidad logarítmica disminuye bastante.

La tercera secuencia, aun teniendo observaciones 1 no pasa al estado oculto 3 ya que las cuatro siguientes observaciones pasarían a otros estados ocultos, y del estado oculto 3 a otros no puede pasar, por lo que es más probable que al principio tenga estados ocultos menos probables, para que al final pueda tener más probables, ya que siendo dos los que puede fallar son cuatro los que puede acertar.

En la cuarta secuencia pasa básicamente por el mismo problema cuando observa el primer valor 1, no puede predecir el estado oculto 3 ya que los siguientes también serian 3, por lo que predice otro estado menos probable, y así permite que el resto sean más probables.

La diferencia de probabilidad de la tercera secuencia de observaciones con la cuarta se debe a que la tercera pasa por alto las dos primeras observaciones mientras que en la cuarta solo una, lo que lo hace más probable.

En el siguiente paso, vamos a comprobar la robustez del algoritmo Viterbi, para esto usaremos la función dhmm_sample, que nos proporcionará una secuencia de estados ocultos ideales y otra de observaciones. Después dejaremos al algoritmo que analice las observaciones y compararemos las dos salidas para ver los errores.

A partir de ocho observaciones distintas obtenemos estos estados ocultos. Iremos aumentando el tamaño de las observaciones para ver cómo evolucionan los errores.

Ocho secuencias de longitud 8

Sample	11222333	13322211	13311323	13111333	12222111	13322222	11112222	11333333
Viterbi	11113333	22222222	13333333	22222333	22222213	33333333	12222222	11222333

Tabla 8: Observaciones y salida de longitud 8 del caso base

Tenemos las secuencias reales establecidas por el Sample (función que genera una secuencia de observables a partir del modelo) y las estimadas por el algoritmo Viterbi. La tasa de aciertos media de las estimaciones del Viterbi es de 51,5%

T_f = ((3+5+3+5+3+6+3+3)/8)/8 = 48.5% por lo que la Ta = **51.5%**

Ocho secuencias de longitud 20

Sample	13332121223333333222	11333333333332333332	13331333333333333333	11211111122222221233
Sample	13332211333333133333	11133332222122113111	12133331122222222221	12222222233111222221
Viterbi	33322221133333333333	13333333333333222222	12223331122111333112	12222222222222222222
Viterbi	33332222213333333333	22222223333333333333	22233333333333333122	22222221333331113333

Tabla 9: Observaciones y salida de longitud 20 del caso base

La tasa de aciertos media se mantiene estable con un 52,5%, la cual es algo baja.

T_f = ((7+9+7+16+5+15+11+6)/8)/20 = 47.5% por lo que la Ta = **52.5%**

Ocho secuencias de longitud 50

En este caso, se aprecia un aumento en la tasa de aciertos que es de un 61%, vamos a analizar los errores con una matriz de confusión.

Sample: Path1	123112222233332133331111322213322222333333322233l2
Sample: Path2	133222223333312333333223332222113331312333333221113
Sample: Path3	113333333111121221133133322212212222222121233132231
Sample: Path4	133333111333333331333333333333333333333333333111121
Sample: Path5	123333333222333332122111113312311122222222222222221
Sample: Path6	121231112122222222223333333331333333312221122221l
Sample: Path7	122211331333333222333222221121133311112333333322ll
Sample: Path8	123333322212222122331333333333333311333331323322333
Viterbi: Path1	113333222211333333333333333333322222233333333333333
Viterbi: Path2	333111222221111133333333333333333333333333222222222
Viterbi: Path3	111113331111111111133312222222222222222222222222222
Viterbi: Path4	333333333333333333333333333333333333333333333312222
Viterbi: Path5	133333333333333333332222222222222233333333333311113
Viterbi: Path6	222222222222222222221333333333333333333333311222222
Viterbi: Path7	122213333333333333332222211111333311111333332222222
Viterbi: Path8	333333222222222222233333333333333333333333222222223

Tabla 10: Observaciones y salida de longitud 50 del caso base

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,06	0,085	0,08
Nubes (2)	0,035	0,175	0,1025
Sol (3)	0,0275	0,06	0,375

Tabla 11: Matriz de confusión salida de longitud 50 del caso base

Ocho secuencias de longitud 100

En este caso, se aprecia un aumento en la tasa de aciertos que es de un 62.5%. Vamos a analizar los errores con una matriz de confusión. Aquí es donde obtenemos un tamaño adecuado para medir correctamente el error, y así estabilizarlo.

Sample:	113332113222223231121333311133333311123333321113332231133333333323333112212 3
Path1	333331333222213333311121
Sample:	132122333332212222211113122233222333333222332222333333333333333333333322133
Path2	33221332222333222233332
Sample:	121222132133333331333331233331111322111112222333333333333332222321133333311 32
Path3	22222131233333333333333
Sample:	111333322333322111222333332222233333223332211333311111112222233333333311112 2
Path4	11333313333332211333332
Sample:	122311122122233122222332222111131111133322222222233323333222213223233222333 3
Path5	32332122123333333212233 3
Sample:	113112222331111133331113333333322211211222112333333222233333322233333321222 3
Path6	33333332221333311222333 3
Sample:	122222233333332221333333332133333333331221332222233333333221332331313333333 3
Path7	33333333333331331331133
Sample:	133333333222221222233333222223323123333222233112212122222333332221111222222 1
Path8	22222222222233133333333 1
Viterbi:	133311222222223333333333333333333333333333333333322222113333333333333111111 3
Path1	33333333322211333331112 0
Viterbi:	131111333322222221333333333333333333333333333122222233333333333333333333333 3
Path2	33333333333333111333333
Viterbi:	111133333333333333333333333331111333331122213333333333333322222213333322222
Path3	22133333333333333333333
Viterbi:	222333333333331222222333333333333333333222222333222221133333333333331122222
Path4	22333333333333333333333
Viterbi:	133333333333333333333333333331222223333332222222223333333333333333333333333
Path5	33333333333333332221133
Viterbi:	222222213311111333311133333333333111122111133333333333333333333333333333333
Path6	33333333333333333333
Viterbi:	222211333333333333333333333333332222222222222223333333311133333333333333333
Path7	33333333333333333333333
Viterbi:	221333332222222222222333333333333333322222222222222211133333332222222222 2
Path8	22222222222133333333333

Tabla 12: Observaciones y salida de longitud 100 del caso base

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,05125	0,05875	0,07750
Nubes (2)	0,03875	0,11625	0,15375
Sol (3)	0,00500	0,04125	0,45750

Tabla 13 Matriz de confusión salida de longitud 100 del caso base

La tasa de aciertos es de un 62.5%. Como podemos ver, hay un mayor número de aciertos cuando el día es soleado, esto se debe a que cuando el día t esta soleado hay un 80% que el día t+1 esté soleado. Es más sencillo de predecir al igual que hay un 60% de que se repita nubes cuando ya ha habido nube, por lo que hay más nubes que lluvia que solo tiene un 40% de probabilidad de que se repita.

La matriz de transiciones influye considerablemente en los resultados, al igual que la matriz de observaciones, pero en este caso las observaciones no favorecen ningún estado en concreto con las probabilidades, y es la matriz de transiciones lo que decanta la balanza.

Ocho secuencias de longitud 500

La tasa de aciertos es de **64.775%,** apreciando una pequeña mejora.
La matriz de confusión tiene unos datos muy parecidos a la anterior.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,04375	0,0485	0,085
Nubes (2)	0,03025	0,0825	0,1255
Sol (3)	0,011	0,052	0,5215

Tabla 14: Matriz de confusión salida de longitud 500 del caso base

Ocho secuencias de longitud 1000:

La tasa de aciertos es de **64.7375%**.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,042	0,059875	0,0915
Nubes (2)	0,028625	0,114875	0,11075
Sol (3)	0,012	0,049875	0,4905

Tabla 15: Matriz de confusión salida de longitud 1000 del caso base

Ocho secuencias de longitud 2000

La tasa de aciertos es de **65.125%**.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,03825	0,0575625	0,0849375
Nubes (2)	0,0265	0,128375	0,1163125
Sol (3)	0,011875	0,0515625	0,484625

Tabla 16: Matriz de confusión salida de longitud 2000 del caso base

Ocho secuencias de longitud 5000

La tasa de aciertos es de **64.495%**.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,04015	0,05755	0,08255
Nubes (2)	0,028775	0,117725	0,1244
Sol (3)	0,011625	0,05015	0,487075

Tabla 17: Matriz de confusión salida de longitud 5000 del caso base

6.1. Evolución del error

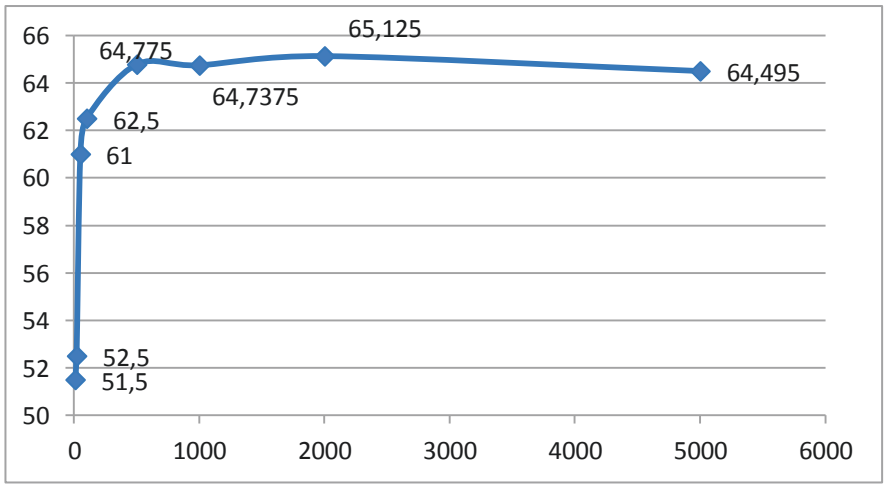


Figura 6: Evolución del error en el problema hmm_tiempo

Como hemos podido observar, el algoritmo Viterbi no mejora el resultado en función de la longitud de la secuencia, aunque es cierto que necesita un tamaño mínimo para poder estabilizarse y sacar un buen resultado. Es complicado disminuir más el error, ya que la matriz de probabilidad de transiciones es poco determinista y con una misma observación las probabilidades de que se encuentre en un estado oculto o en otro son muy similares.

Probaremos a modificar las matrices de observaciones para ver si podemos disminuir el error en el algoritmo Viterbi.

Acciones / Tiempo	ANDAR (1)	COMPRAR (2)	LIMPIAR (3)
LLUVIA (1)	10%	10%	80%
NUBES (2)	10%	80%	10%
SOL (3)	80%	10%	10%

Tabla 18: Matriz de probabilidades de observación aumentando el determinismo del modelo del caso base

Con esta nueva matriz de probabilidades, vamos a obtener una nueva matriz de confusión y analizar los errores.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,12515	0,016925	0,042325
Nubes (2)	0,02325	0,198425	0,0442
Sol (3)	0,025475	0,02985	0,4944

Tabla 19: Matriz de confusión del aumento del determinismo del caso base

Observando la matriz de confusión vemos un claro aumento de los aciertos para la clase 1 y 2, por lo que la tasa de aciertos aumenta a un 81,8%. Con esto observamos una clara relación entre la matriz de probabilidades y la tasa de acierto del modelo.

Ahora utilizaremos una matriz de observaciones en la que sea prácticamente aleatorio el estado oculto en función de dichas observaciones.

Acciones / Tiempo	ANDAR (1)	COMPRAR (2)	LIMPIAR (3)
LLUVIA (1)	33%	33%	33%
NUBES (2)	33%	33%	33%
SOL (3)	33%	33%	33%

Tabla 20: Matriz de probabilidades de observación aleatorizando el modelo del caso base

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0	0	0,1785
Nubes (2)	0	0	0,277775
Sol (3)	0	0	0,543725

Tabla 21: Matriz de confusión aleatorizadas las observaciones del caso base

Como recordamos, hay un mayor porcentaje de estados de sol, ya que cuando se pasa por el estado 3, la probabilidad de que este se repita es de un 80%. Teniendo en cuenta que en el modelo las observaciones no ofrecen información sobre los estados al ser las probabilidades iguales, clasifica todos los estados al 3, ya que éste es el estado más probable.

Para concluir, modificaremos las transiciones entre estados para que ninguno de estos tenga preferencia. Con esta modificación, damos preferencia a que se repita el estado en que se encuentra pero a ninguno de los tres en concreto.

T / T+1	LLUVIA (1)	NUBES (2)	SOL (3)
LLUVIA (1)	60%	20%	20%
NUBES (2)	20%	60%	20%
SOL (3)	20%	20%	60%

Tabla 22: Matriz de transición completamente aleatorizada del caso base

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,3336	0	0
Nubes (2)	0,33345	0	0
Sol (3)	0,33295	0	0

Tabla 23: Matriz de confusión aleatorizadas el modelo completo del caso base

Al ser prácticamente aleatorio, clasifica todo a un sólo estado, y de entre ellos escoge el mejor estado, es decir, el estado que más aparece en la secuencia.

Por último, vamos a pasar un modelo a la función *sample* para generar las secuencias y otro distinto a la función Viterbi para que estime los estados ocultos. En teoría, este modelo debería ser completamente erróneo. Utilizaremos el modelo inicial para generar los ejemplos, y cambiaremos por completo el modelo que le pasaremos al Viterbi.

Predicción / Real	Lluvia (1)	Nubes (2)	Sol (3)
Lluvia (1)	0,063075	0,108175	0,0089
Nubes (2)	0,17625	0,082175	0,01635
Sol (3)	0,3082	0,054975	0,1819

Tabla 24: Matriz de confusión observaciones creadas por otro modelo del caso base

Como podemos ver, hemos engañado al Viterbi y no ha conseguido más de un 33% de aciertos, funcionando mejor de lo esperado, pero aun así con un porcentaje muy bajo de aciertos.

7. Introducción al problema

Tendremos distintos ficheros con datos simulados con distintos grados de ruido y distintos tamaños de resolución, y además otros en los que se ha usado el filtro de Kalman para utilizar la velocidad en el HMM.

La resolución tiene el mismo efecto que en las imágenes, cuanto mayor es, más numero de cuadrados o de celdas hay, por lo que son más pequeños, y al tener cada cuadrado un color, el hecho de haber más cuadrados hace que haya una mayor capacidad para mostrar tonalidades en la imagen, y por tanto mejora la calidad de la misma.

En este caso, cuando tenemos resolución quiere decir que los puntos que tenemos han sido discretizados en celdas. Entonces cuanto más grande sea el tamaño de la celda, un punto que en realidad no ocupa el mismo espacio que otro puede ocuparlo. Esto es muy similar a realizar aproximaciones. En numerosas ocasiones, cuando hablemos de aumentar la resolución, nos referimos a aumentar el tamaño de las celdas.

El problema de la resolución suele acrecentarse en las curvas, ya que el avión disminuye su velocidad y los sensores consiguen puntos más cercanos entre ellos, por lo que estos pueden superponerse al aumentar la resolución.

El ruido también será un problema, ya que la diferencia angular entre dos puntos puede modificarse ampliamente por el ruido como podremos ver.

El filtro de Kalman tendrá un efecto muy positivo con grandes niveles de ruido, pero también tiene otros problemas como veremos luego.

Por último usaré un fichero con datos reales para probar la eficacia del modelo construido.

7.1. Manual de uso para la aplicación

Tendremos tres ficheros de ejecución dependiendo los datos que queramos clasificar. Si queremos clasificar trayectorias simuladas con distintos grados de ruido o resolución, usaremos el fichero ***simuladasHMM.m***. Si queremos clasificar trayectorias simuladas a las que se les ha aplicado un filtro de Kalman previamente y además tienen distintos grados de ruido o resolución usaremos el fichero ***simuladasKalmanHMM.m***. Finalmente si queremos clasificar trayectorias reales, independientemente si se les ha aplicado filtro de Kalman usaremos el fichero ***realesHMM.m***. Sabiendo esto vamos a dividir la explicación en cinco partes, y todos los ficheros tendrán la misma estructura, aunque no serán idénticos.

Elección de parámetros

Las cabeceras de los ficheros tienen el mismo formato; tienen una zona de "parámetros" donde según los datos que introduzcas obtendrás los datos de entrada que has pedido para que el algoritmo los clasifique, y en función de estos usaremos un modelo u otro, ya que no es posible clasificar de la misma manera, por ejemplo, datos con un ruido mínimo, que con un mayor ruido, y al igual pasa con el filtro de Kalman.

- Tenemos el fichero ***simuladasHMM.m*** para los datos simulados con resolución o con ruido.

```
r=5; % 1-5 Grado de Resolucion o Ruido
nr=1;%resolucion 1 || ruido 2
track=1;%1,2,3
sensor=1;
```

- Tenemos el fichero ***simuladasKalmanHMM.m*** para los datos simulados con resolución o con ruido a los que se ha aplicado el filtro de Kalman.

```
r=5; % 1-5 Grado de Resolucion o Ruido
nr=1;%resolucion 1 || ruido 2
track=1;%1,2,3
sensor=1;
```

- Tenemos el fichero ***realesHMM.m*** para los datos reales.

```
filtro=0;% sin filtro 0 || con filtro 1
trayectoria=3;%1-4
Nsensor=1;
```

Explicación de los datos

Las trayectorias simuladas con las que vamos a trabajar son las siguientes:

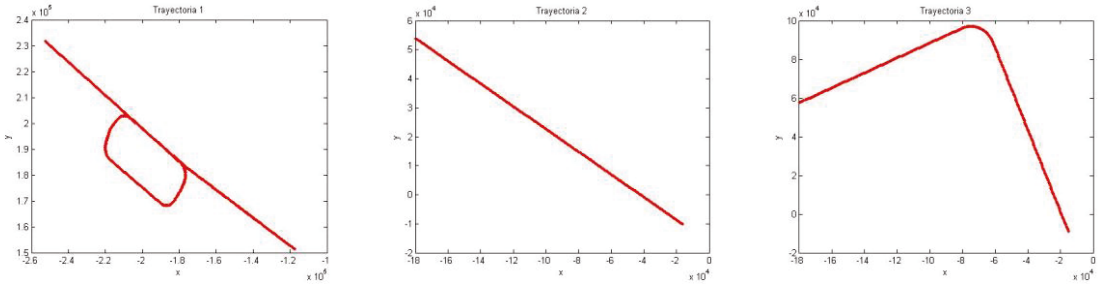


Figura 7A,B,C:Trayectorias simuladas

Como podemos ver, la primera trayectoria es un hipódromo, formada por un conjunto de giros a derecha y rectas, siendo la trayectoria sólo una vuelta.
La segunda trayectoria es una simple recta.
Y la tercera, es un giro a izquierdas de noventa grados.

También tenemos las trayectorias reales que son las siguientes:

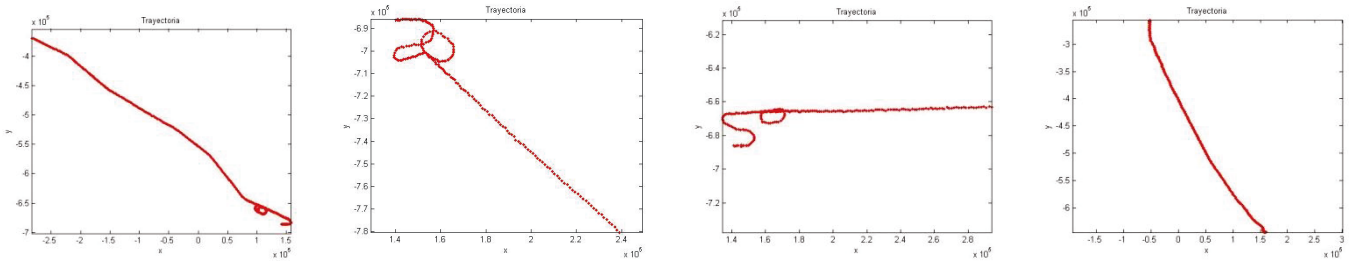


Figura 8A,B,C,D:Trayectorias reales

Tratamiento de los datos o creación de los observables

Para las trayectorias en las que no se ha aplicado el filtro de Kalman, se sacan las posiciones que forman la trayectoria, y de éstas obtendremos las diferencias angulares entre los puntos según la siguiente fórmula:

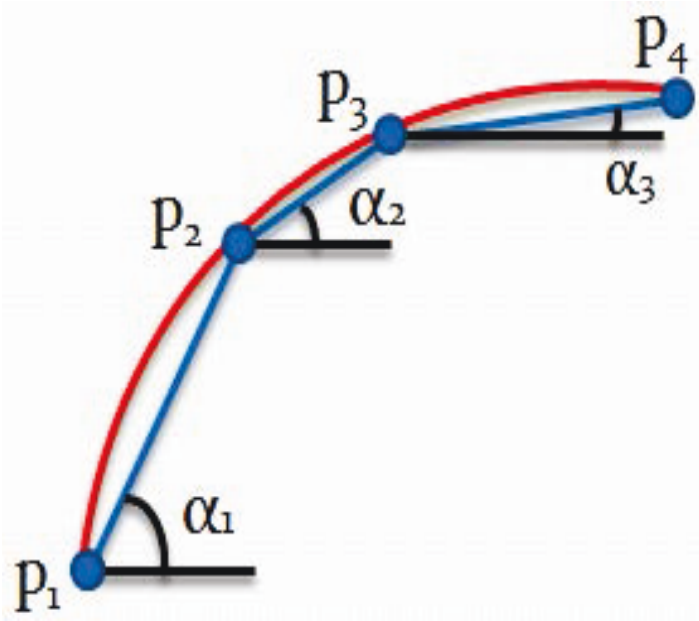


Figura9: Creación de los observables

```
for i=1:(length(A)-1),
 x = (A(i)-A(i+1));   %x = A(i)
 y = (B(i)-B(i+1));   %y = B(i)
 angulo =atan2(y,x)*180/pi;
 C(i)= angulo;
 if (i>1)
    dif=C(i-1)-angulo;
    while(abs(dif-360)< abs(dif))
         dif= dif -360;
    end
    while(abs(dif+360)< abs(dif))
         dif= dif +360;
    end
    D(i-1)= dif;
    D(i)=0;
    D(i+1)=0;
 end
end
```

Figura10: Código de creación de los observables (sin filtro de Kalman)

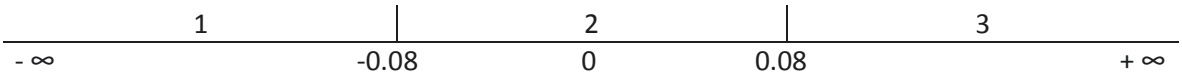
Dentro del bucle *for* que recorre el vector "A" que contiene las posiciones en el eje x de la trayectoria, el B tiene las posiciones del eje y.
Usamos la arco tangente para obtener el ángulo, y tras esto, pasamos de radianes a grados. Cuando tenemos el primer ángulo, a partir de éste realizamos la diferencia con el segundo, para así hallar la diferencia angular de todas las posiciones de la trayectoria con sus siguientes. El relleno del vector D que contiene las diferencias angulares, se produce porque es necesario que sea del mismo tamaño que el de A y B si quiero representarlo en alguna gráfica (no son del mismo tamaño porque para obtener el ángulo necesito dos posiciones de A y para la diferencia angular necesito dos ángulos, por eso mismo la longitud es dos valores menor).

Cuando sí hemos aplicado el filtro de Kalman, no usamos la posición para obtener el ángulo, sino que el filtro de Kalman nos proporciona la velocidad angular, y gracias a esto no tenemos que calcular el ángulo. Simplemente realizamos la diferencia angular, debido a que el filtro de Kalman usa el vector velocidad para reducir el ruido en las trayectorias. Omitiendo este detalle, el resto es completamente igual.

El resultado de esto serán diferencias angulares continuas, y posteriormente las discretizaremos para convertirlas en los observables. Está claro que la diferencia angular será mayor entre puntos cuanto mayor sea el giro, y cuando el avión vaya recto las diferencias angulares serán 0 o cercanas a 0, esa es la relación entre los observables y los estados ocultos. Después se las pasaremos al algoritmo y este estimará las acciones en cada momento.

La discretización de las diferencias angulares para convertirlas en observaciones será de tres tipos:

1. Diferencia angular alejada de cero y negativa.
2. Diferencia angular cercana a cero.
3. Diferencia angular alejada de cero y positiva.



Las líneas marcan los topes en la discretización, el tope entre 1 y 2 es de -0.08 grados, mientras que el que está entre 2 y 3 es de 0.08 grados.
Los topes en la discretización irán cambiando en función de los cambios en los datos de entrada para generar los observables.

Elección del modelo

La elección del modelo viene definida por algunos de los parámetros elegidos al principio, dependiendo del fichero se escogerán entre resolución o ruido, el grado de resolución o ruido, o si se usan datos a los que se ha aplicado un filtro de Kalman.

En función de esto, se busca un modelo que previamente ha sido entrenado con distintas trayectorias con los parámetros indicados. No se utiliza el algoritmo EM ni similares [10] ya que los resultados obtenidos son peores que entrenando manualmente el modelo a base de prueba y error.

Funcionamiento del algoritmo

Las entradas al algoritmo son el HMM entrenado y los observables discretizados, una vez tiene estos datos el algoritmo Viterbi realiza la clasificación en los posibles estados de los datos.

Las acciones o estados que el algoritmo dará como resultado son las siguientes:

- Girar a la izquierda.
- Recto.
- Girar a la derecha.

Evaluación de la salida

Los datos simulados, además, cuando no usemos los ideales, serán evaluados mediante una matriz de confusión, que conseguiremos clasificando la trayectorias ideales, ya que en estas no se comente ningún error.

Con esto, conseguiremos el porcentaje de aciertos y errores en cada uno de los estados posibles, para poder evaluar el resultado en detalle.

8. Modelo para datos simulados ideales (sin ruido y sin resolución)

Para utilizar la herramienta aplicada al problema, primeramente utilizaremos datos obtenidos mediante un simulador sin ruido y sin resolución, debido a que el modelo debe funcionar mejor cuando los datos no tienen ruido o factores que modifiquen su trayectoria. Así podremos ir adaptando el HMM cada vez para que los observables sean los más reales posibles y mejorar la salida del algoritmo.

El fichero de datos incluye varias trayectorias que provienen de diferentes sensores. En este caso, el sensor no tiene importancia ya que nos darán las mismas posiciones, cogeremos solo una al azar para las diferentes trayectorias.

Para este primer caso, la matriz de observación y de transiciones serán las siguientes:

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.65	0.25	0.1
Recto (2)	0.1	0.8	0.1
Giro Dcha. (3)	0.1	0.25	0.65

Tabla 25: Matriz de transición del modelo para datos ideales

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.55	0.4	0.05
Diferencia angular pequeña (2)	0.2	0.6	0.2
Diferencia angular grande positiva (3)	0.05	0.4	0.55

Tabla 26: Matriz de observación del modelo para datos ideales

Trayectoria 1

Vamos a trabajar sobre la trayectoria 1 (*figura 7A*). Primero obtendremos una gráfica con las diferencias angulares, esta gráfica nos permite ver donde se encuentran estas diferencias, es muy útil para saber donde debemos introducir los topes para la discretización.

Una vez que las hemos discretizado se muestran en la gráfica de la derecha, las diferencias angulares discretizadas son los observables que pasaremos al algoritmo.

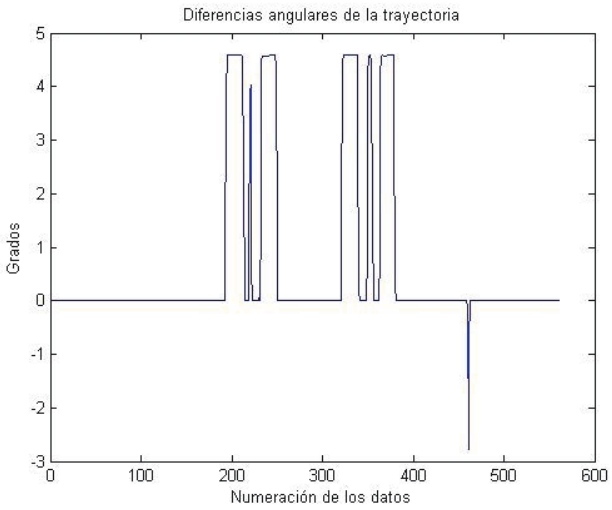


Figura 11A

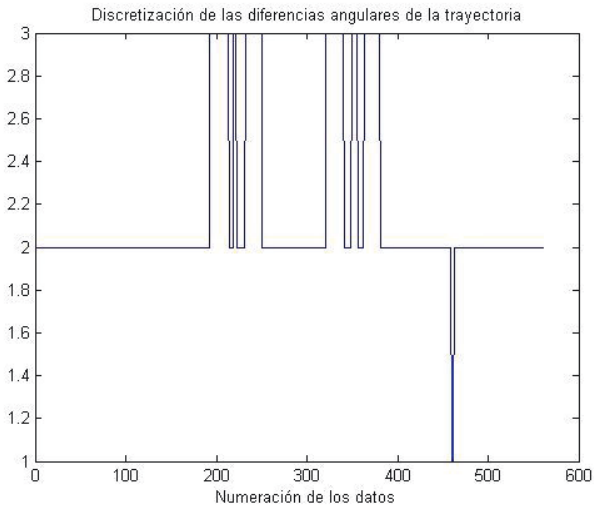


Figura 11B

Una vez tenemos los observables y el modelo preparados, se lo pasamos al algoritmo y nos devuelve la salida. Como podemos ver en el gráfico, es una buena estimación de las acciones realizadas.

La trayectoria no tiene giros a izquierda, por lo que no ha clasificado ningún dato a 1, mientras que los giros a derechas (valor 3) han sido correctamente clasificados.

Como hemos mencionado antes, las rectas se clasifican como 2.

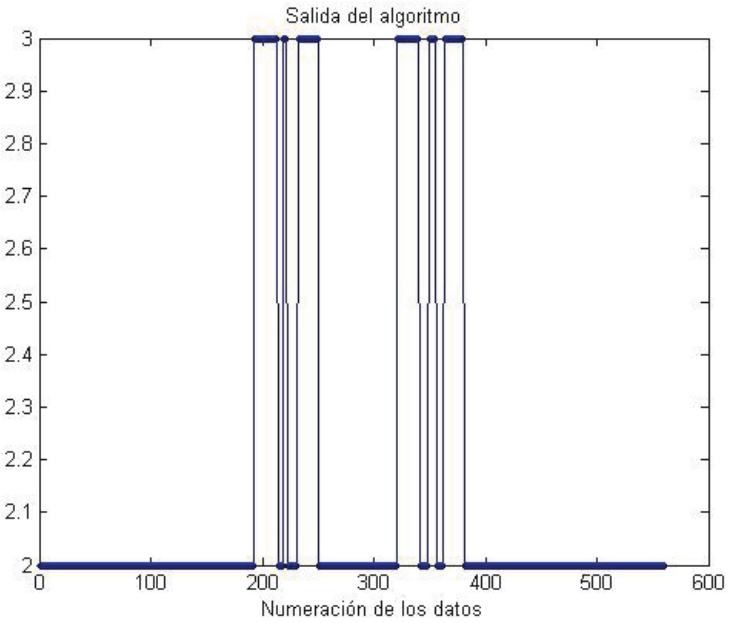


Figura 11C

El siguiente gráfico muestra la misma información, pero representada de una manera distinta, y además nos muestra la trayectoria pudiendo así apreciar la clasificación de las acciones que saca el algoritmo. Es decir, el eje X y el eje Y muestra la trayectoria, mientras el Z muestra si está girando a la derecha (3) si va recto (2) o si gira a la izquierda (1). Así podremos ver rápidamente si el algoritmo clasifica bien o no, ya que si vemos que la trayectoria tiene un giro y se encuentra en el valor 2 del eje Z estaría mal clasificado.

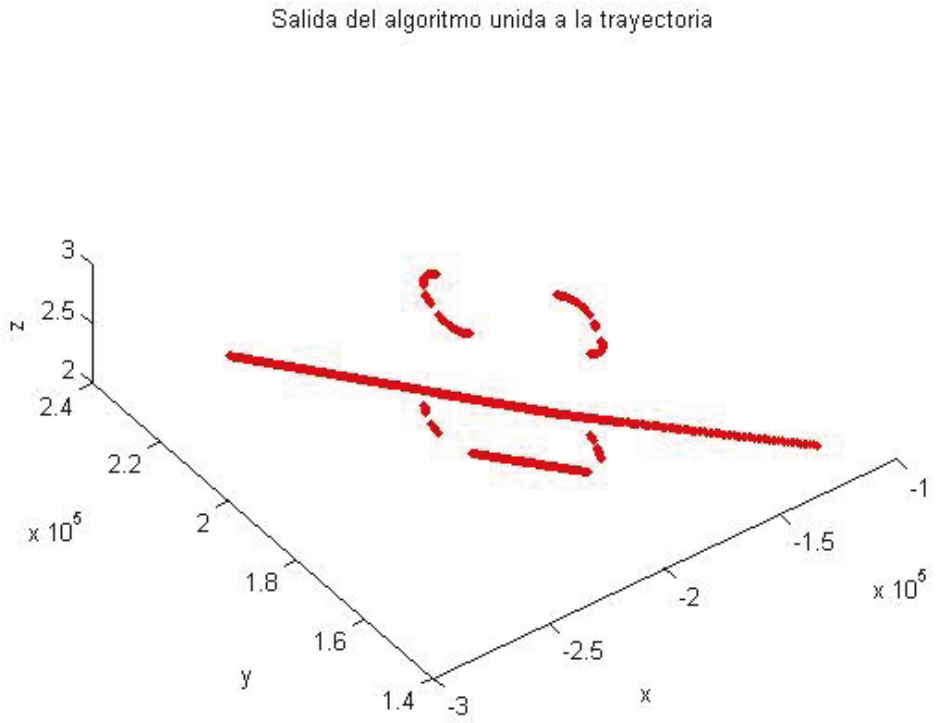


Figura 11D

Trayectoria 2

Vamos a trabajar sobre la trayectoria 2 (*figura 7B).*

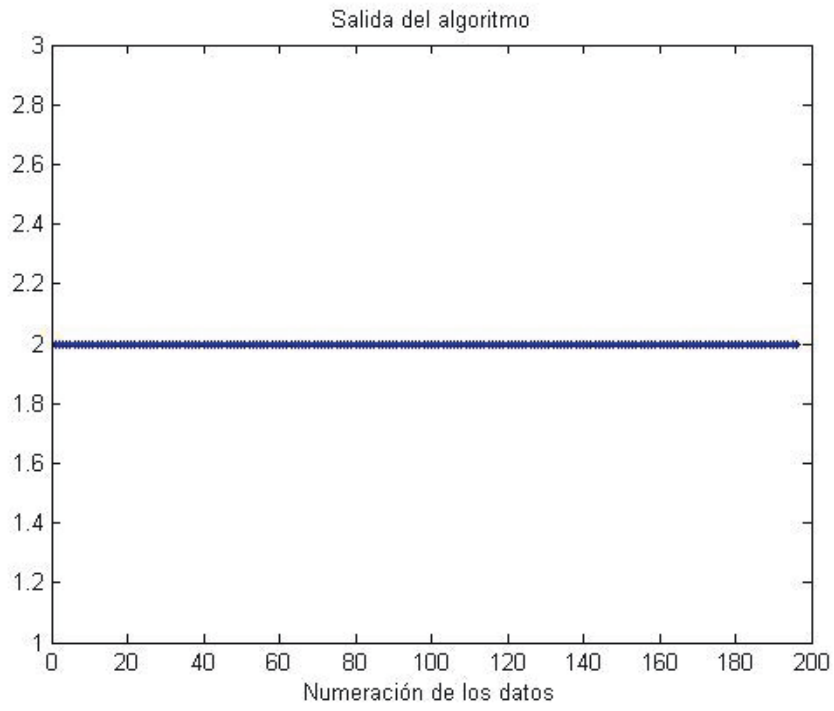


Figura 12A

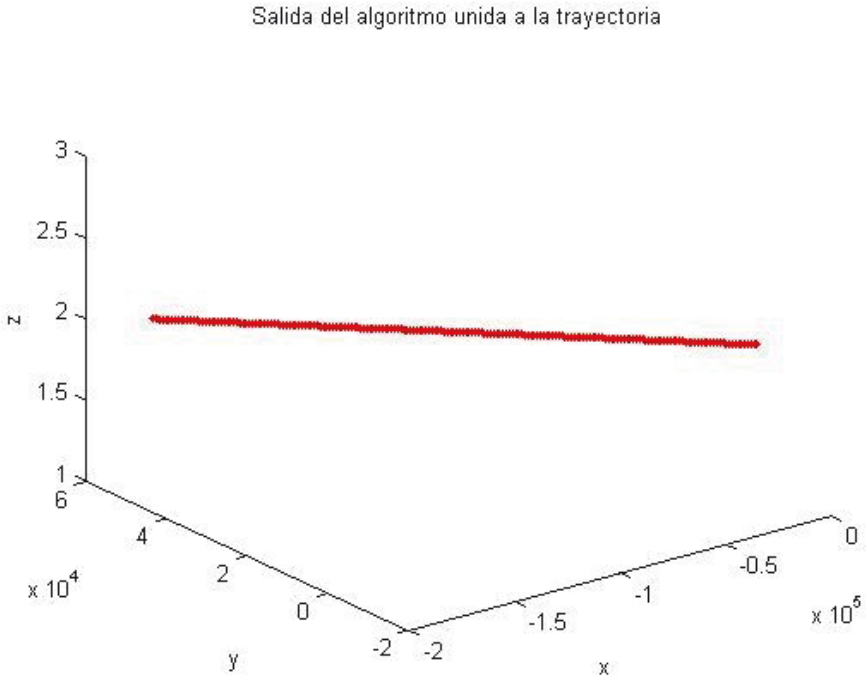


Figura 12B

Trayectoria 3

Vamos a trabajar sobre trayectoria 3 (*figura 7C*),

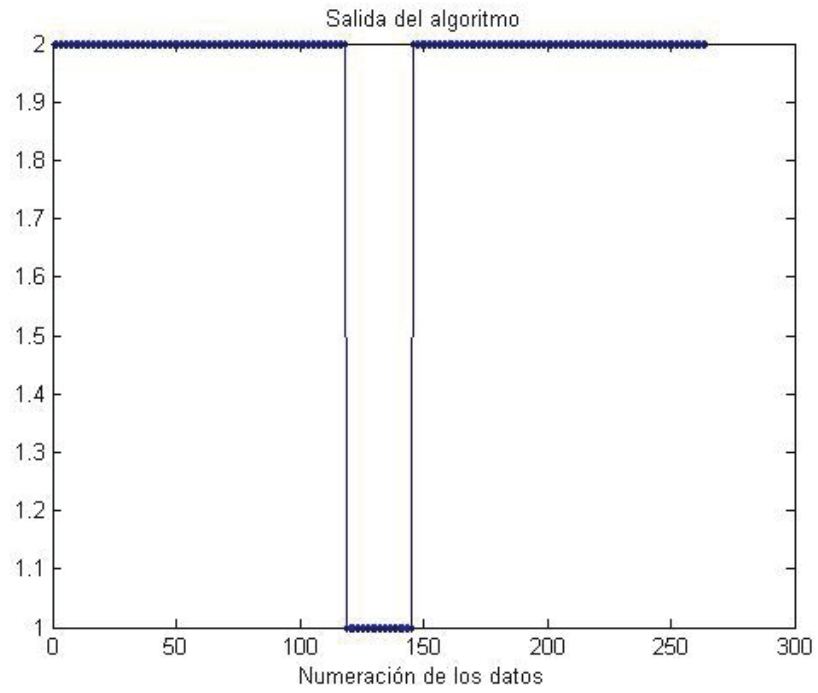


Figura 13A

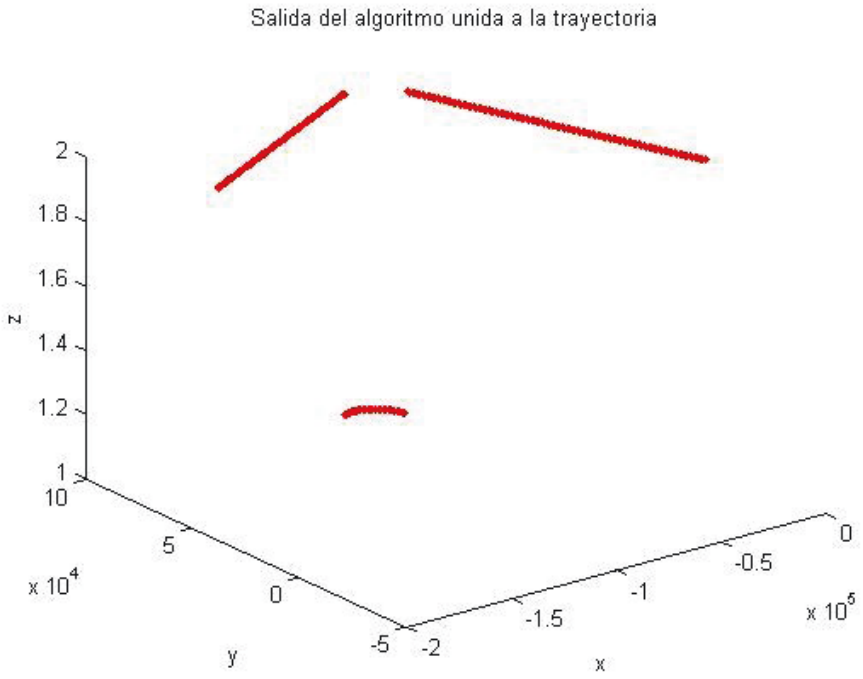


Figura 13B

9. Modelo con datos simulados con resolución y sin ruido

Para los datos simulados con resolución, tenemos varios ficheros con distintos grados de resolución, así podremos analizar cómo puede afectar esto al algoritmo y hasta qué punto puede suplir sus efectos.

El problema en este caso, es que los ficheros con resolución discretizan las posiciones obtenidas por los radares en pequeños espacios, entonces en rectas cuando las velocidad es mayor es razonable, pero en curva la distancia recorrida es menor, y los puntos son más cercanos, incluso se pueden encontrar en la misma celda, lo que dificulta la labor del algoritmo, ya que las entradas no son las esperadas.

Empezaremos con ligera resolución e iremos aumentando ligeramente. Se cojera en el simulador un valor nominal de resolución y se multiplicará por 1/8, 1/4, 1/2, 1 y 2. Debemos tener en cuenta que el problema angular del sensor, que produce que a mayor distancia del sensor, mayor será el tamaño de la celda, por lo que cuanto más lejos se encuentre, peores resultados obtendremos.

Para este caso, hemos modificado las matrices de transiciones y de observaciones del modelo, ya que los datos que tenemos no son totalmente fiables por la resolución, y también irémos modificándola con los distintos grados.

9.1. Datos con 1/8 de resolución.

La matriz de transiciones y la de observaciones serán las siguientes:

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.65	0.25	0.1
Recto (2)	0.1	0.8	0.1
Giro Dcha. (3)	0.1	0.25	0.65

Tabla 27: Matriz de transición del modelo para datos con 1/8 de resolución

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.55	0.4	0.05
Diferencia angular pequeña (2)	0.15	0.7	0.15
Diferencia angular grande positiva (3)	0.05	0.4	0.55

Tabla 28: Matriz de observación del modelo para datos con 1/8 de resolución

Trayectoria 1

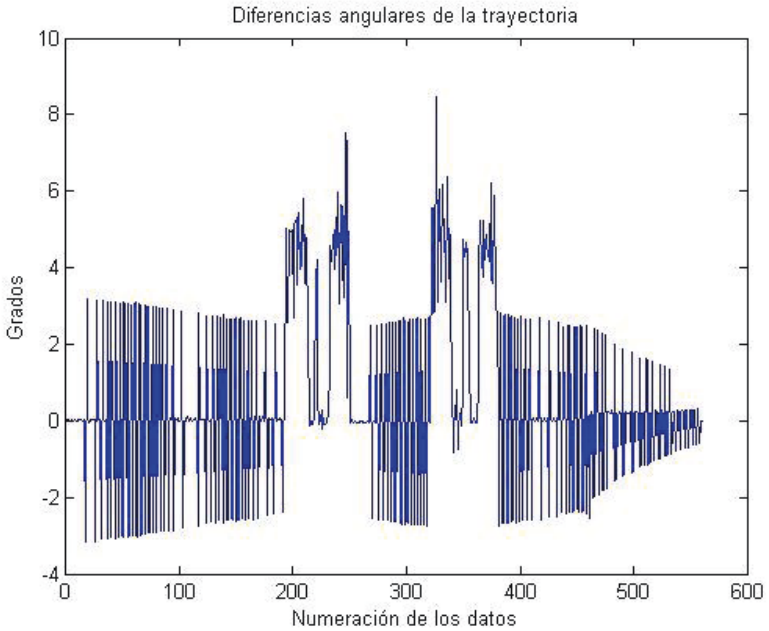


Figura 14A

Como podemos ver en la figura 14A, necesitamos aumentar el rango entre los topes que antes se encontraban en -0.08 y 0.08, a 3.5 y -3.5, ya que necesitamos discretizar correctamente los datos para que el modelo funcione.

1	2	3
- ∞ -3.5	0	3.5 + ∞

Una vez discretizamos las diferencias angulares, obtenemos la figura 14B.

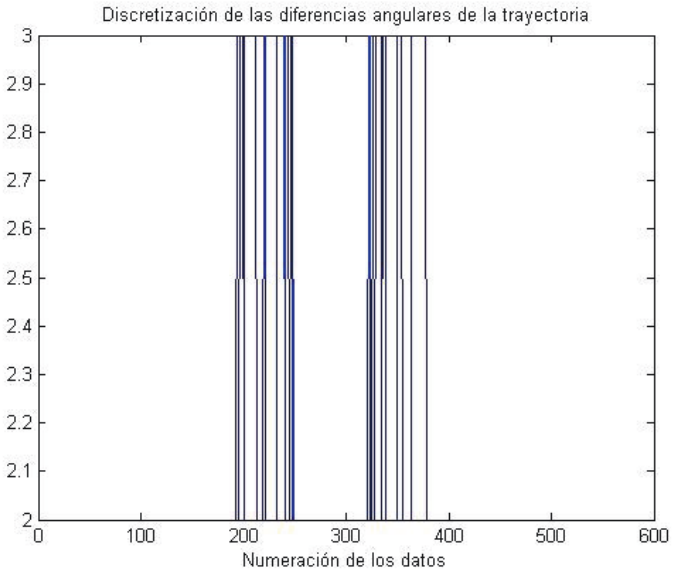


Figura 14B

No es tan exacta como la discretización que obtenemos de los datos ideales, pero con estos datos se los introducimos al modelo y debe encargarse de mejorar los resultados, como podemos apreciar en los resultados.

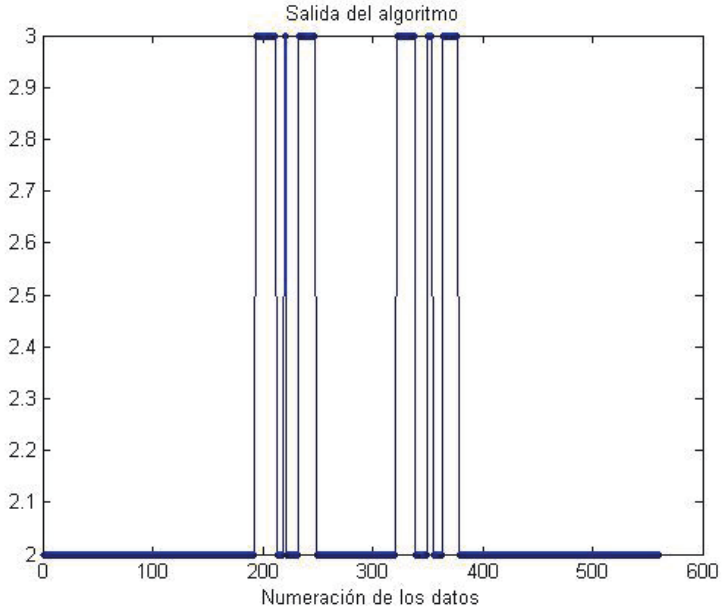


Figura 14C

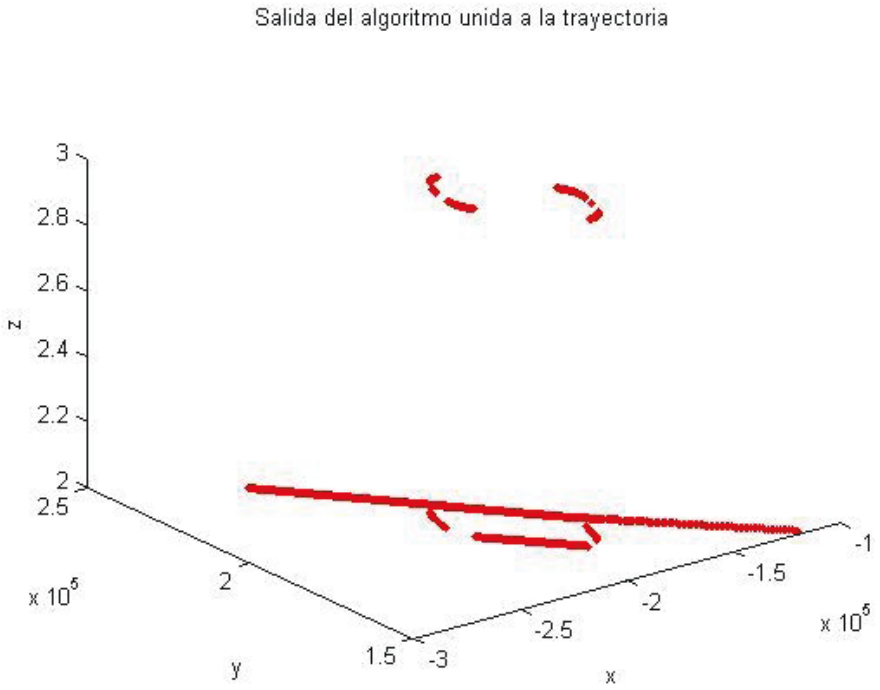


Figura 14D

Para analizar la eficacia del algoritmo, usaremos una matriz de confusión, así podremos saber donde realiza los errores de clasificación.

Los datos ideales los hemos calculado con la trayectoria anterior (sin ruido y sin resolución).

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0	0,835714285714286	0
G. Dcha. (3)	0	0,0285714285714286	0,130357142857143

Tabla 29: Matriz de confusión de la salida con 1/8 de resolución para trayectoria 1

Como podemos ver, el algoritmo acierta en la mayoría de los casos, y no tienen tendencia a cometer ningún error en concreto, los pocos errores que existen son mínimos, no llegando ni a un 3%.

Trayectoria 2

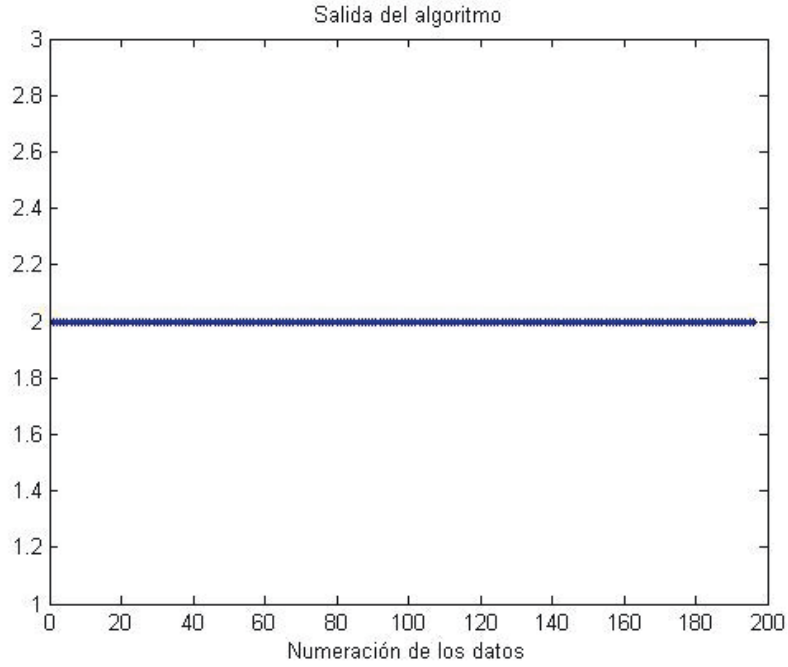


Figura 15A

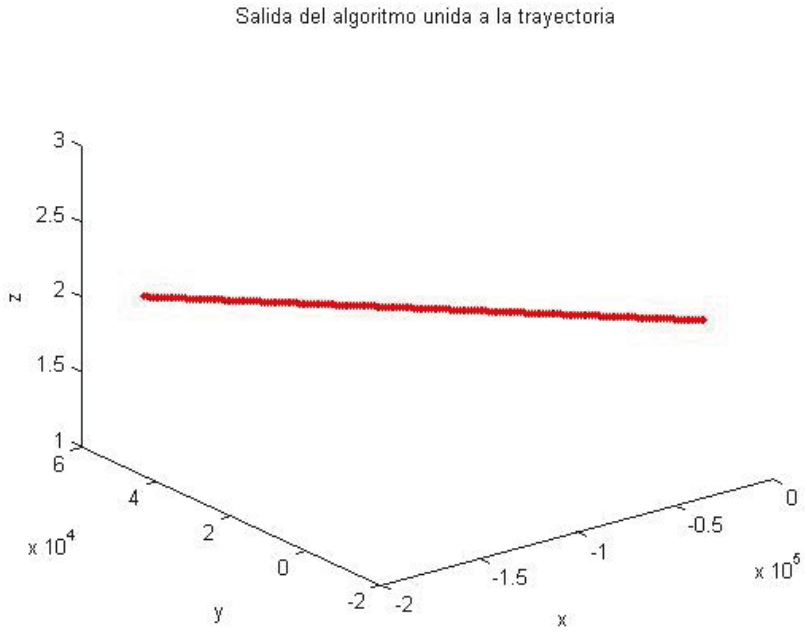


Figura 15B

Como ya vemos en las figuras 15A y 15B, no tiene ningún error. Esto se debe a que la resolución no es muy elevada y la trayectoria no tiene ningún giro, lo que facilita el trabajo al algoritmo.

Trayectoria 3

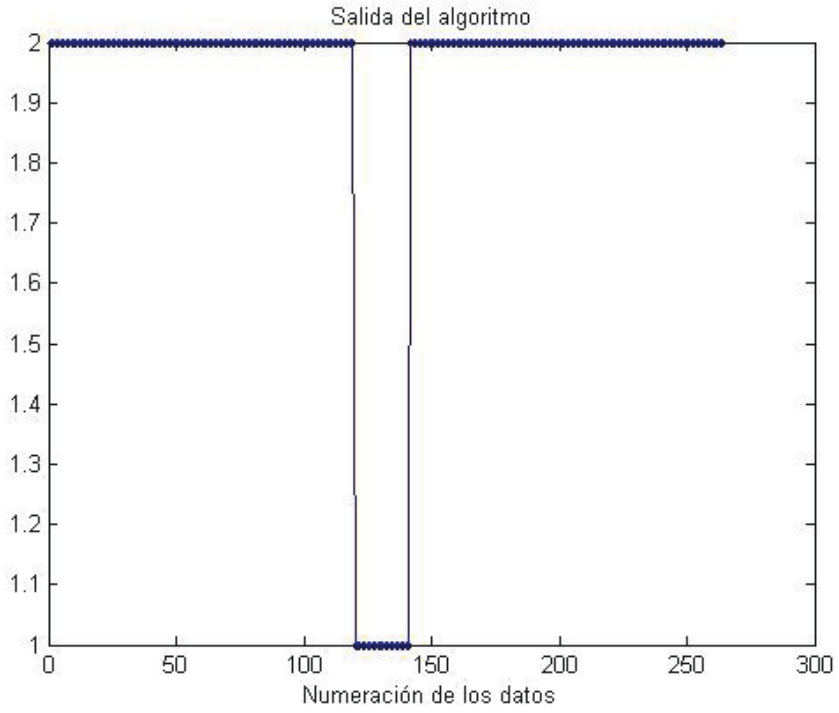


Figura 16A

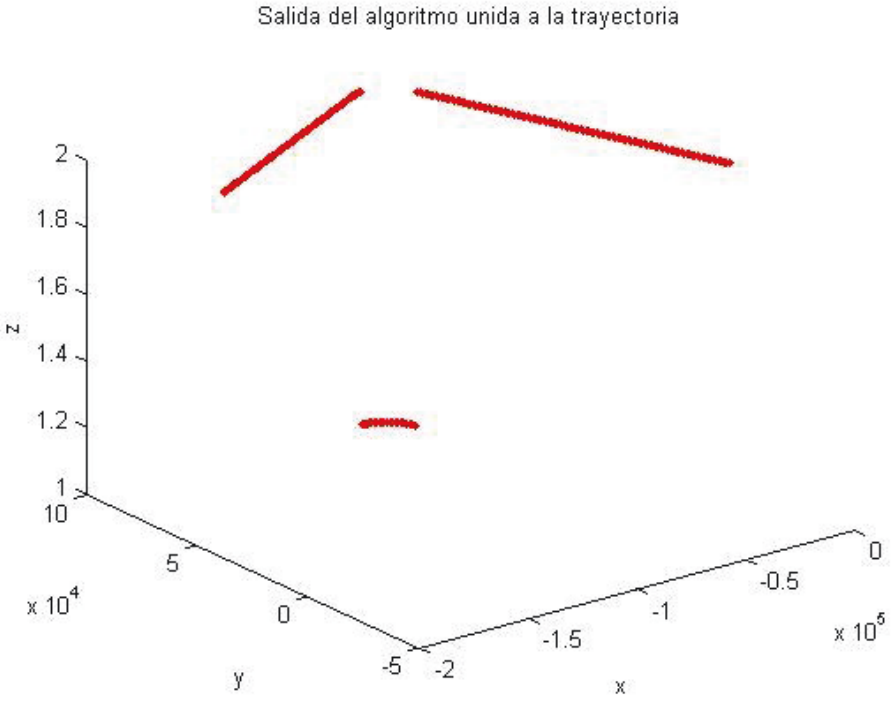


Figura 16B

Para esta última trayectoria, la matriz de confusión ofrece un resultado muy similar a las anteriores:

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0833333333333333	0,0189393939393939	0
Recto (2)	0	0,897727272727273	0
G. Dcha. (3)	0	0	0

Tabla 30: Matriz de confusión de la salida con 1/8 de resolución para trayectoria 3

Como hemos podido ver, el algoritmo funciona perfectamente con este tamaño de resolución, para la siguiente prueba probaremos a aumentarlo hasta 1/4 y observaremos los resultados.

9.2. Datos con 1/4 de resolución.

Trayectoria 1

En este caso, la resolución es mayor, lo que aumentará la dificultad al HMM para que funcione correctamente. No obstante, vamos a observar las diferencias angulares de los datos en la figura *17A.*

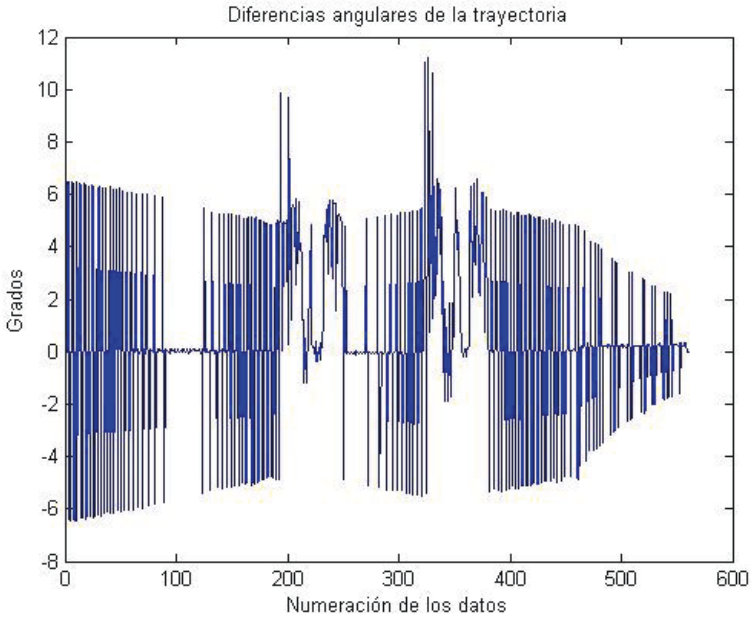


Figura 17A

Como podemos ver, las diferencias angulares se reducen al final, esto quiere decir que la distancia al sensor está influyendo, lo que complicará las cosas a la hora de discretizar los datos y más tarde. Y además, obligará a aumentar el rango entre los topes de discretización, pero no demasiado para no perder información.

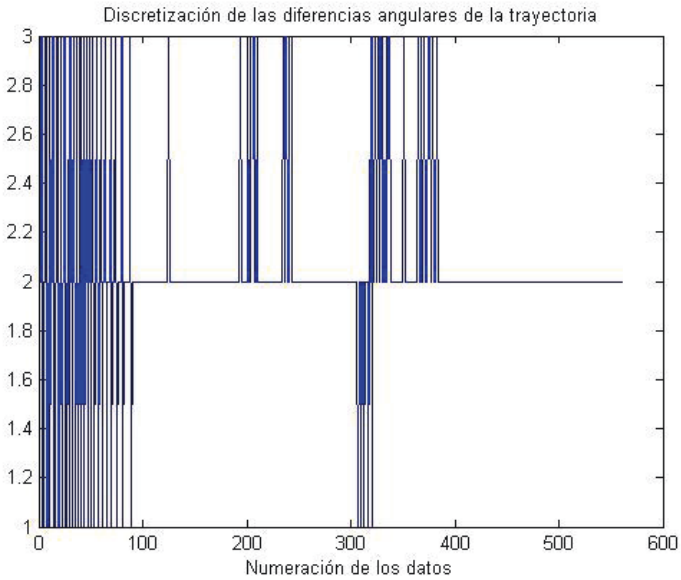


Figura 17B

Como vemos, al ser la resolución muy elevada no podemos evitar que en la discretización salgan estos resultados, pero al final de la trayectoria, podemos ver como mejora, beneficiando así al modelo.

Una vez ajustados los parámetros de las matrices de observaciones y de transiciones expuestas, a continuación podremos ver los resultados del modelo.

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.6	0.25	0.15
Recto (2)	0.1	0.8	0.1
Giro Dcha. (3)	0.15	0.25	0.6

Tabla 31: Matriz de transición del modelo para datos con 1/4 de resolución

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.45	0.44	0.01
Diferencia angular pequeña (2)	0.1	0.8	0.1
Diferencia angular grande positiva (3)	0.01	0.44	0.45

Tabla 32: Matriz de observación del modelo para datos con 1/4 de resolución

También se han cambiado los topes para la discretización a 5.4 y -5.4 para ajustarse al ruido.

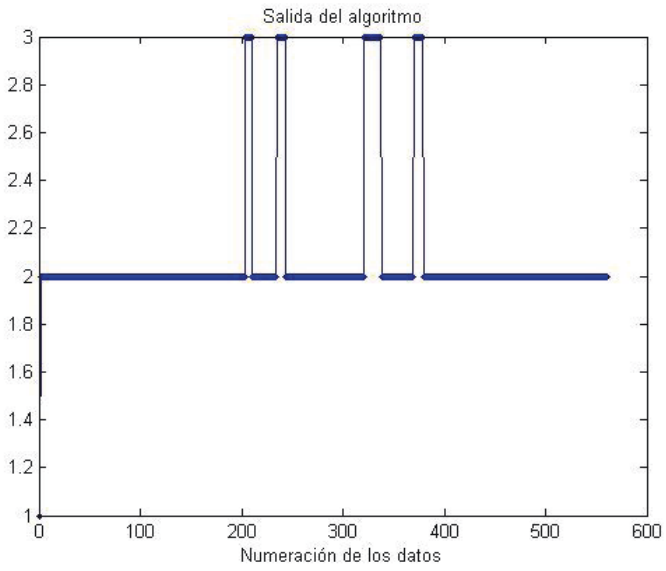


Figura 17C

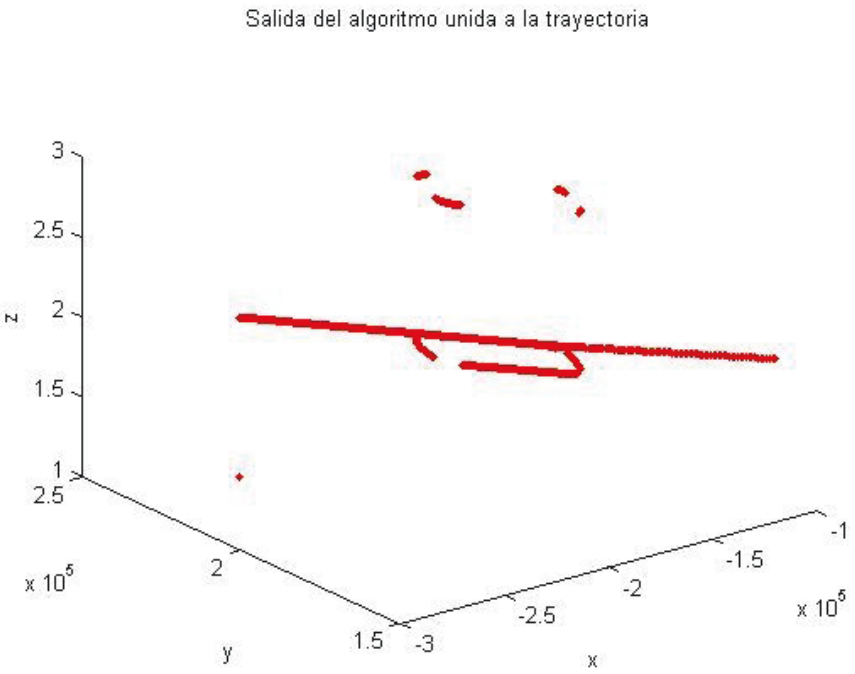


Figura 17D

Como podemos observar, ha perdido precisión en comparación con la prueba realizada con 1/8 de resolución, aun así los resultados son bastante precisos, ya que las curvas más significativas sigue reconociéndolas.

En los gráficos ya podemos empezar a intuir los errores, pero mediante la matriz de confusión veremos con exactitud dónde se producen.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00178571428571429	0,833928571428572	0
G. Dcha. (3)	0	0,0821428571428571	0,0767857142857143

Tabla 33: Matriz de confusión de la salida con 1/4 de resolución para trayectoria 1

Como podemos ver, el algoritmo tiende a clasificar los estados ocultos a recto, tiene un 8% de error clasificando como recto datos que giran a la derecha, esto se debe a que con mayor resolución se disminuyen las diferencias entre los dos estados.

Trayectoria 2

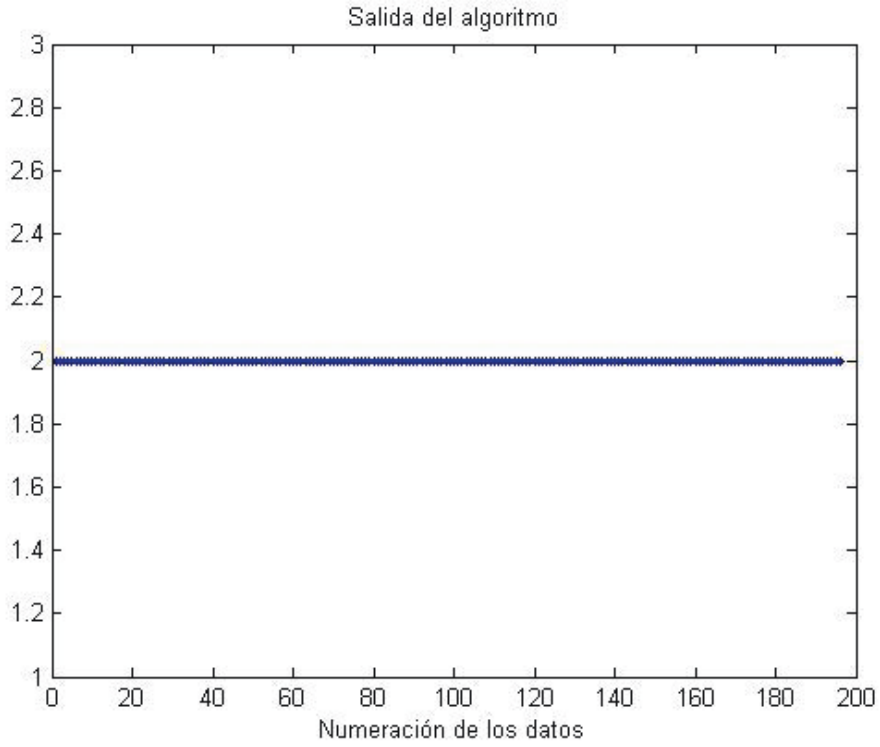


Figura 18A

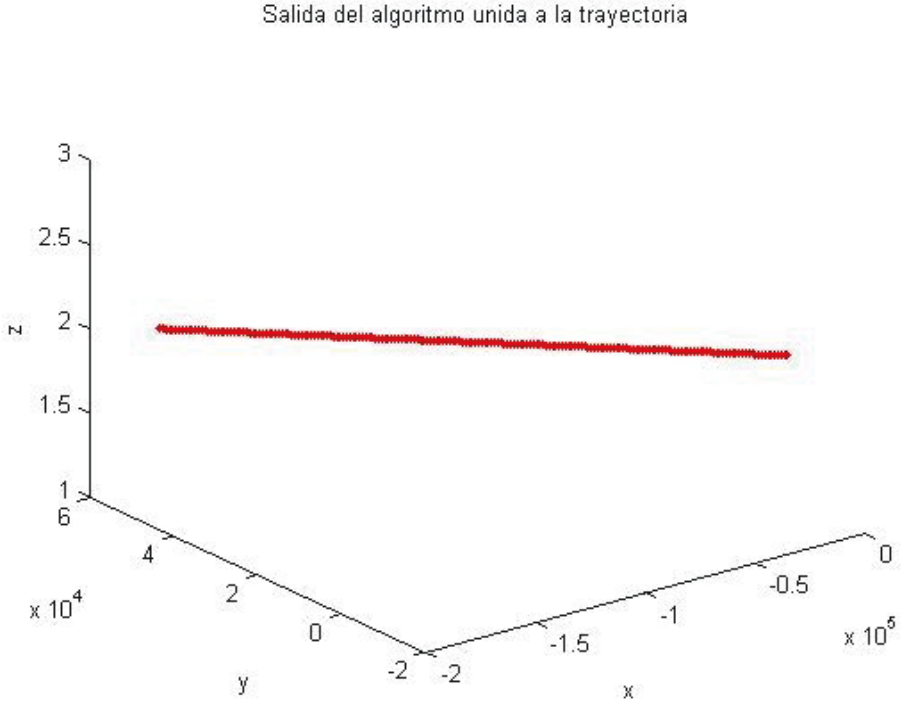


Figura 18B

Trayectoria 3

En este caso vamos a pararnos a analizar la salida del algoritmo para la trayectoria 3. Como recordamos, la trayectoria 3 tenía una curva de 90 grados a izquierdas, pero la salida del algoritmo nos lo muestra sin curva.

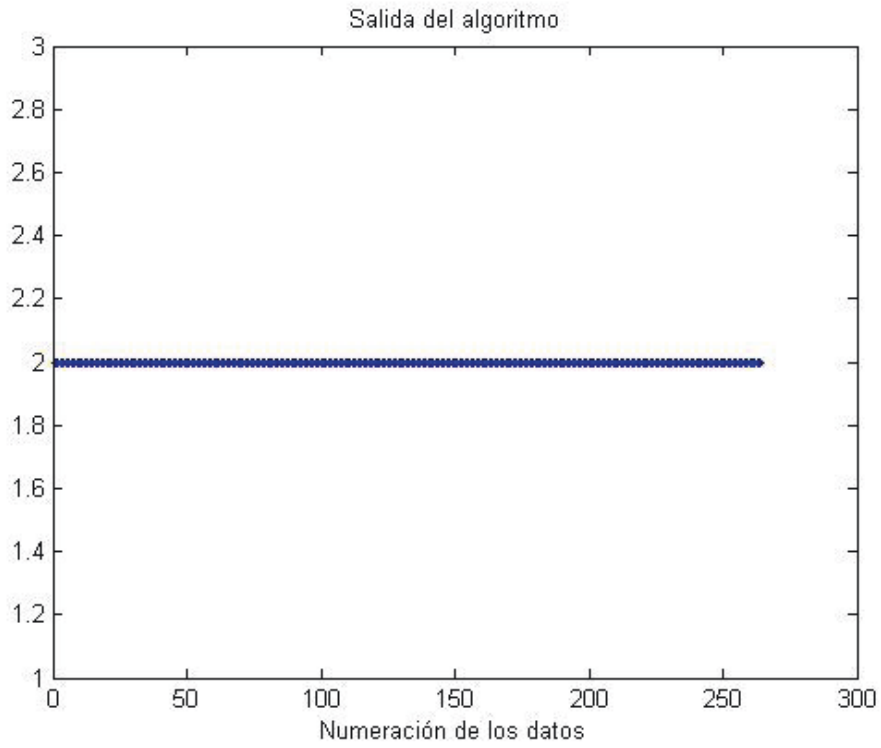


Figura 19A

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,102272727272727	0
Recto (2)	0	0,897727272727273	0
G. Dcha. (3)	0	0	0

Tabla 34: Matriz de confusión de la salida con 1/4 de resolución para trayectoria 3

Esto se debe a un problema en la discretización de las diferencias angulares, los topes utilizados aquí son demasiado altos como vemos en la gráfica de las diferencias angulares.

La diferencia en la curva aumenta, pero no lo suficiente para clasificarlo correctamente ya que el tope esta en -5.4 y no lo supera en ningún caso.

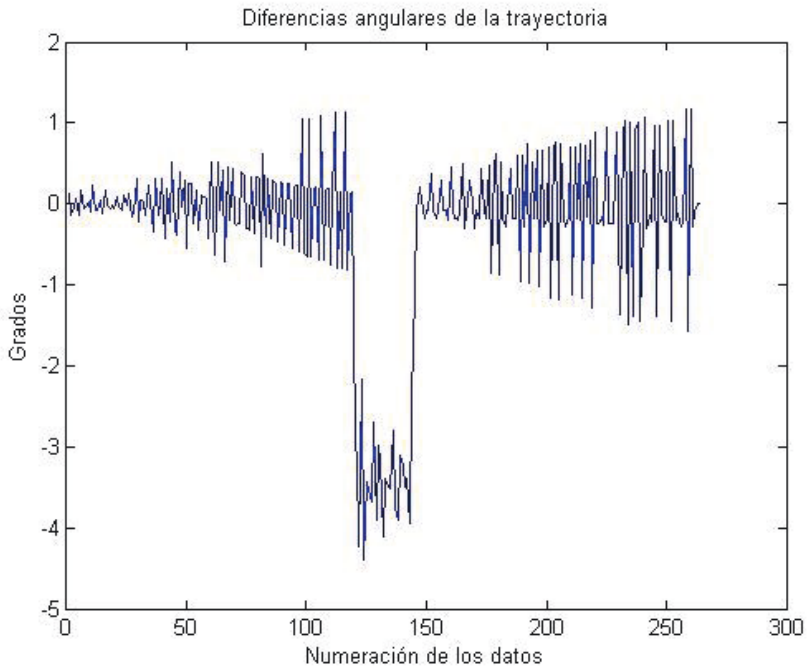


Figura 19B

Esto se debe a la lejanía de la primera trayectoria al sensor, la resolución en esta es mayor, mientras que en la tercera es menor ya que está más cerca.

Si probamos a disminuir el rango entre los topes de 2 a -2, vamos a observar el resultado.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0946969696969697	0,00757575757575758	0
Recto (2)	0	0,897727272727273	0
G. Dcha. (3)	0	0	0

Tabla 35: Matriz de confusión de la salida con 1/4 de resolución para trayectoria 3con corrección de datos

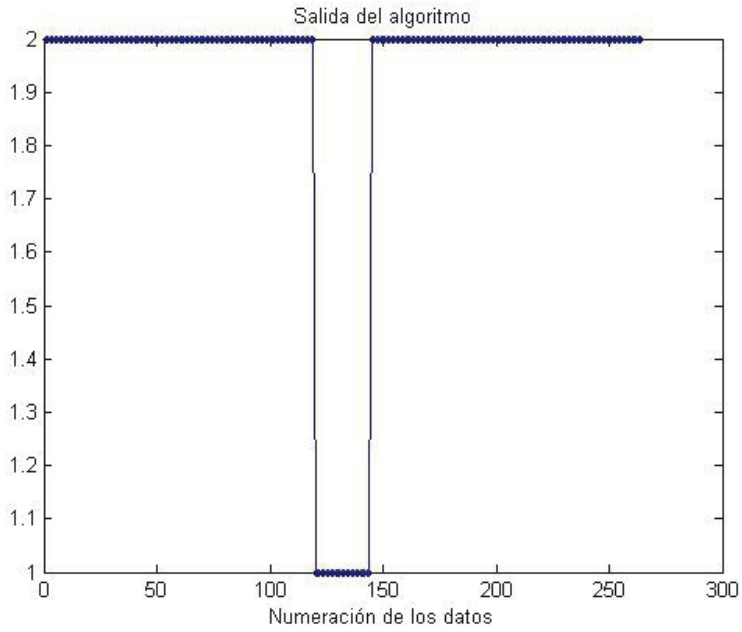


Figura 19C

Con este cambio podemos ver que el error disminuye a menos de un 1%, por eso debemos tener mucho cuidado a la hora de realizar la discretización de los datos.

9.3. Datos con 1/2 de resolución.

Con la resolución tan elevada vamos a ver cómo se deterioran los resultados del algoritmo, y vamos a poder apreciar la importancia de la distancia del sensor a la trayectoria que mide, ya que es un factor clave dado que provoca un aumento en la resolución a medida que se aleja del sensor y esto tendrá un efecto negativo en su salida.

Trayectoria 1

Hemos realizado unos ligeros cambios en el modelo para permitir ajustarse ligeramente mejor a este tamaño de resolución.

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.65	0.25	0.1
Recto (2)	0.1	0.8	0.1
Giro Dcha. (3)	0.1	0.25	0.65

Tabla 36: Matriz de transición del modelo para datos con 1/2 de resolución

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.55	0.4	0.05
Diferencia angular pequeña (2)	0.1	0.8	0.1
Diferencia angular grande positiva (3)	0.05	0.4	0.55

Tabla 37: Matriz de observación del modelo para datos con 1/2 de resolución

Además de modificar los topes de la discretización de 1.5 a -1.5.

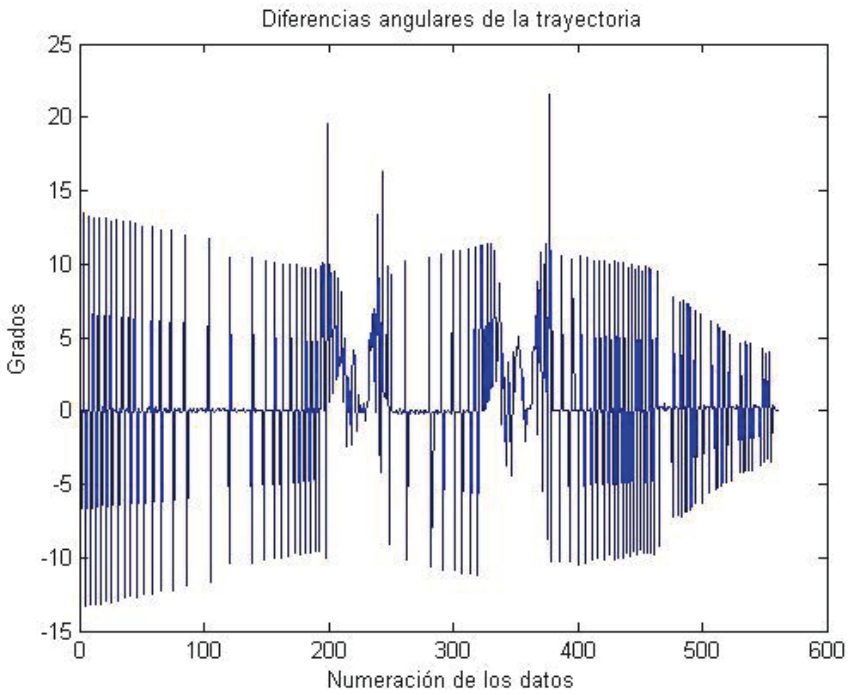


Figura 20A

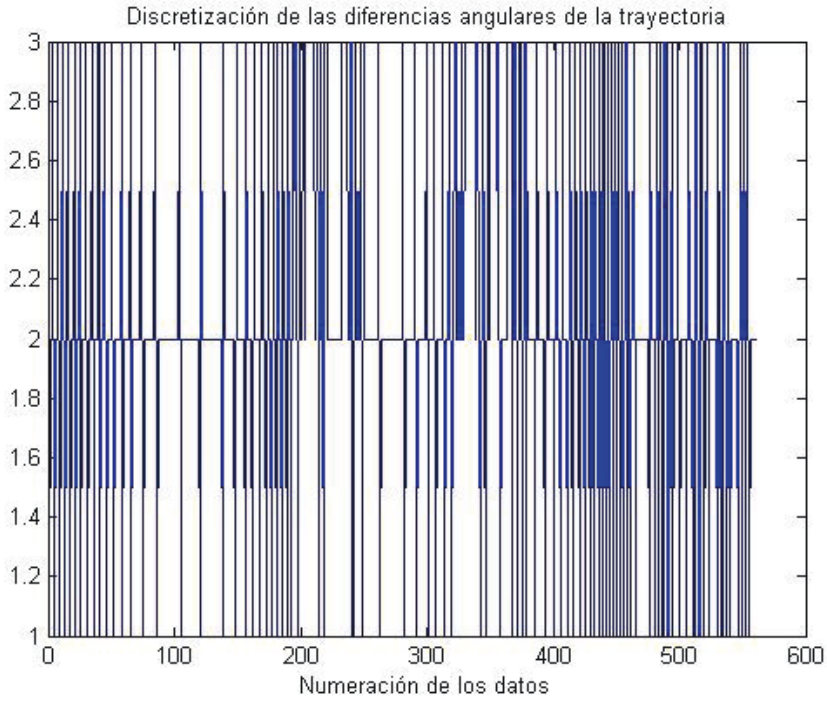


Figura 20B

En las graficas anteriores estamos viendo los inconvenientes de una resolución demasiado alta, la discretización de las diferencias angulares es muy irregular. Esto seguramente afecte de manera negativa a la salida.

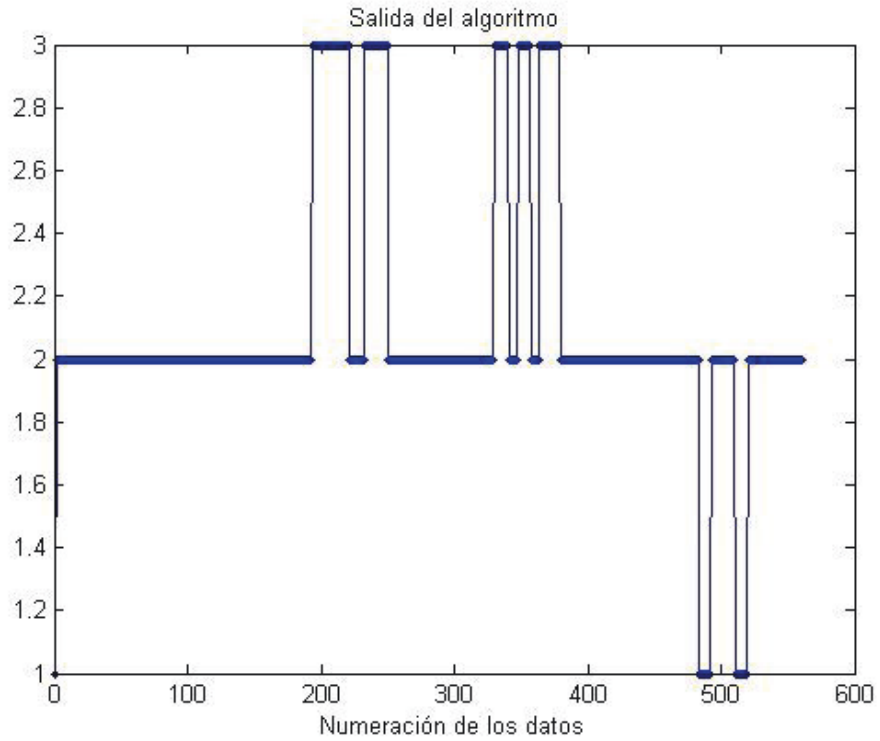


Figura 20C

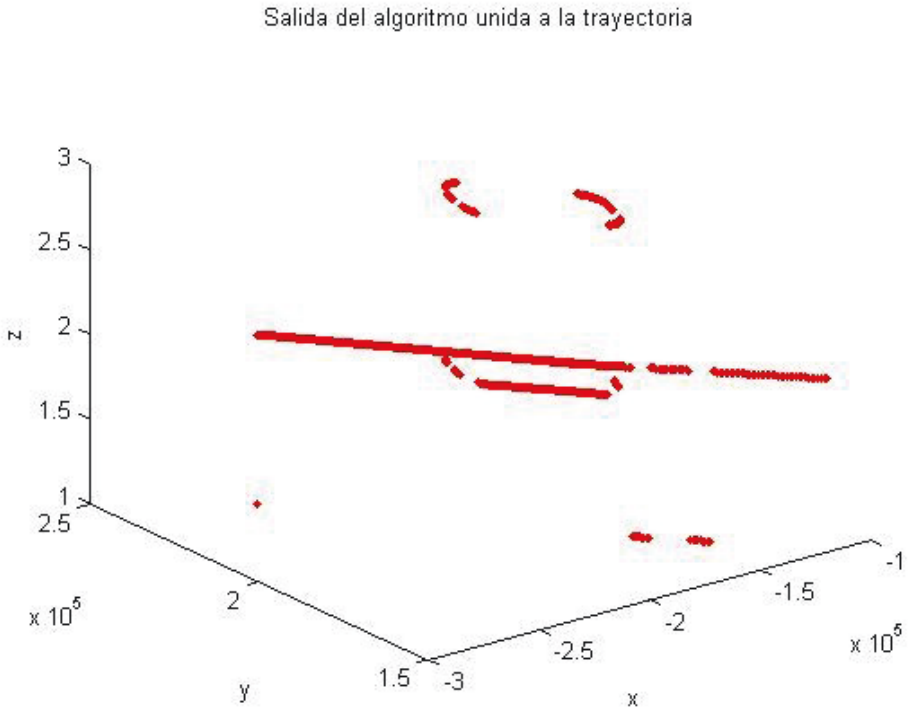


Figura 20D

Hemos podido ver en las gráficas los errores en la salida. No obstante, para un análisis más detallado del porcentaje de los errores y aciertos nos fijaremos en la matriz de confusión.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,0339285714285714	0,787500000000000	0,0142857142857143
G. Dcha. (3)	0	0,0250000000000000	0,133928571428571

Tabla 38: Matriz de confusión de la salida con 1/2 de resolución para trayectoria 1

Como vemos, los errores que más han aumentado son cuando la trayectoria es recta y clasifica como una curva. Esto es un claro efecto del ruido y seguramente bajarían estos errores, aumentando los topes en la discretización pero subirían los de otras celdas.

Trayectoria 2

La trayectoria dos, como recordamos, es una recta. No obstante, en el gráfico inferior podemos observar que las diferencias angulares no dan esa sensación, no son nada constantes y como hemos mencionado anteriormente, esto es el mejor ejemplo de la variación de resolución en función de la distancia al sensor. Esto provocará que en una trayectoria tan sencilla el algoritmo pueda cometer grandes errores.

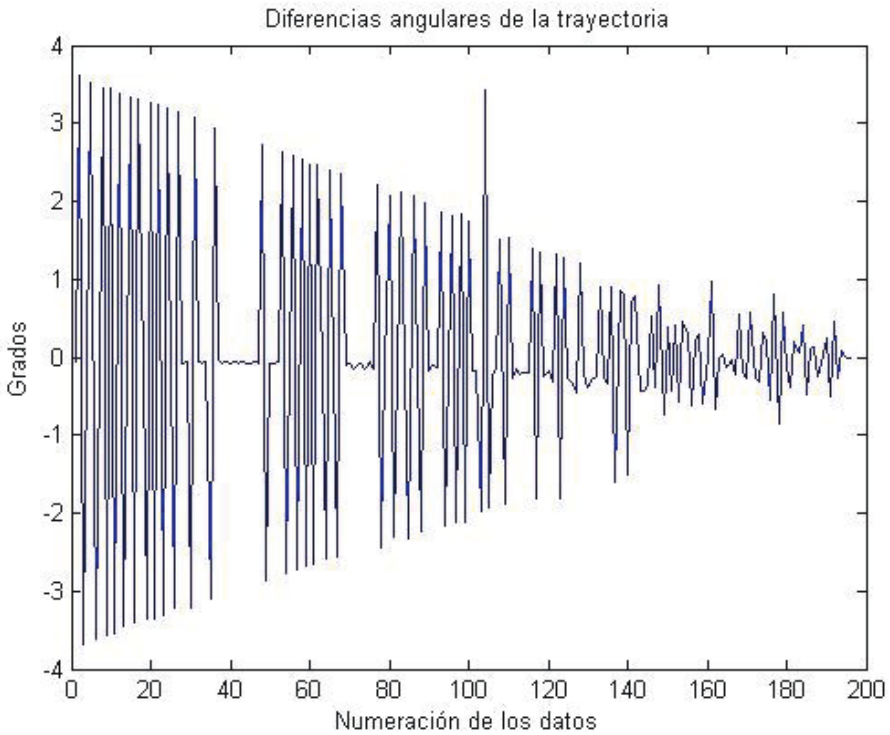


Figura 21A

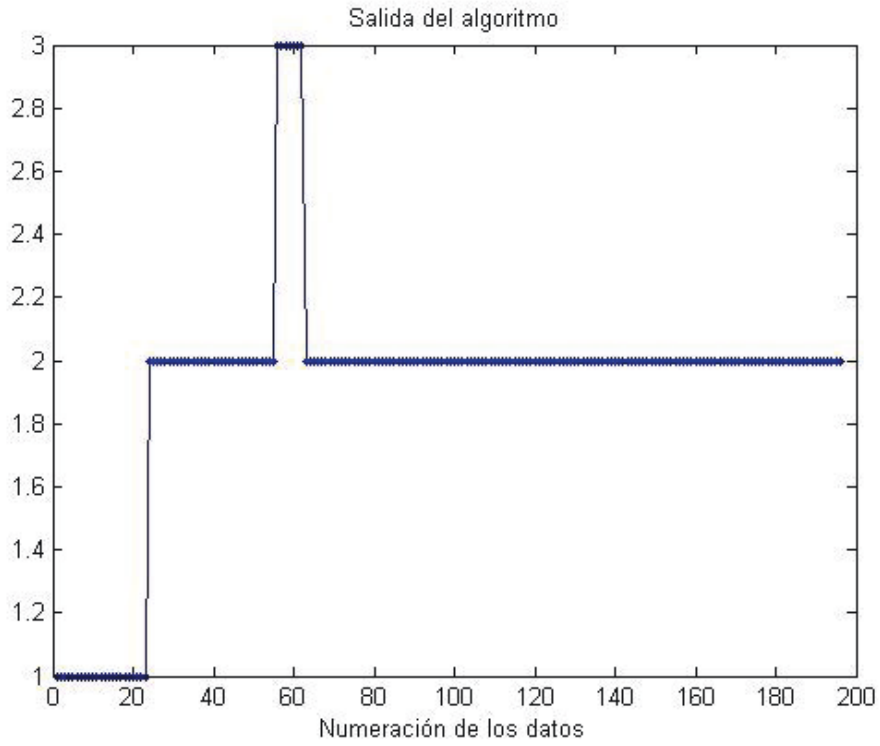


Figura 21B

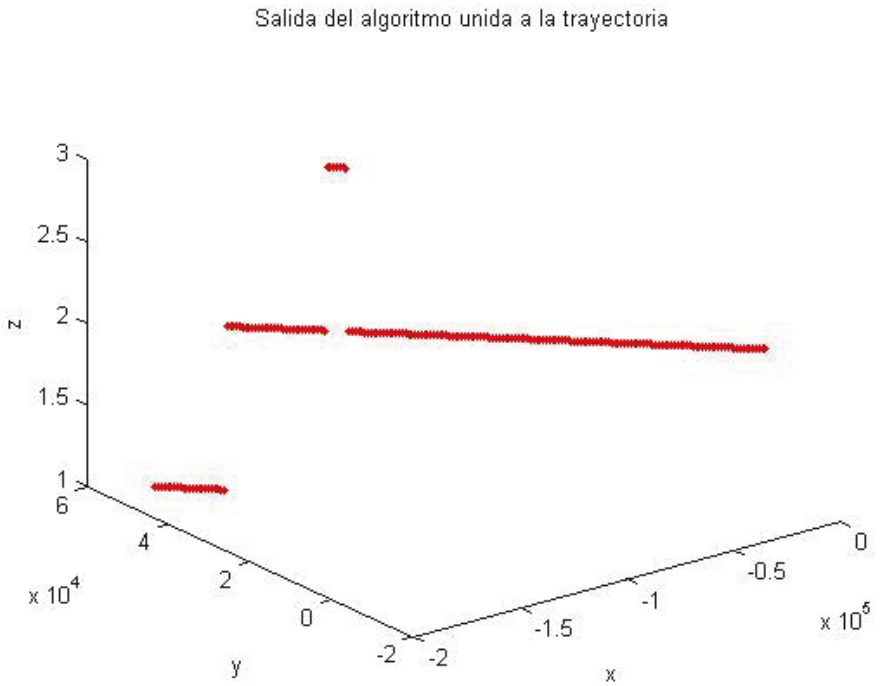


Figura 21C

Como hemos indicado anteriormente, la parte derecha de la trayectoria tiene algunos fallos ya que su tamaño de resolución es mayor ya que se encuentra a más distancia del sensor, mientras que al final o izquierda, más cercana al sensor, se clasifica correctamente.

Trayectoria 3

En este caso tenemos el mismo problema que en la segunda trayectoria, la clasificación falla por el aumento de resolución al final de la trayectoria.

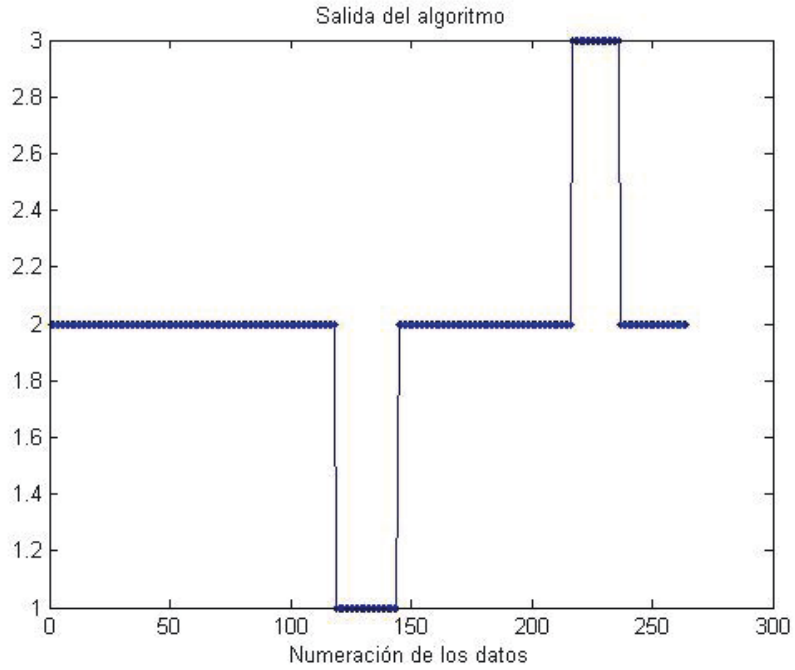


Figura 22A

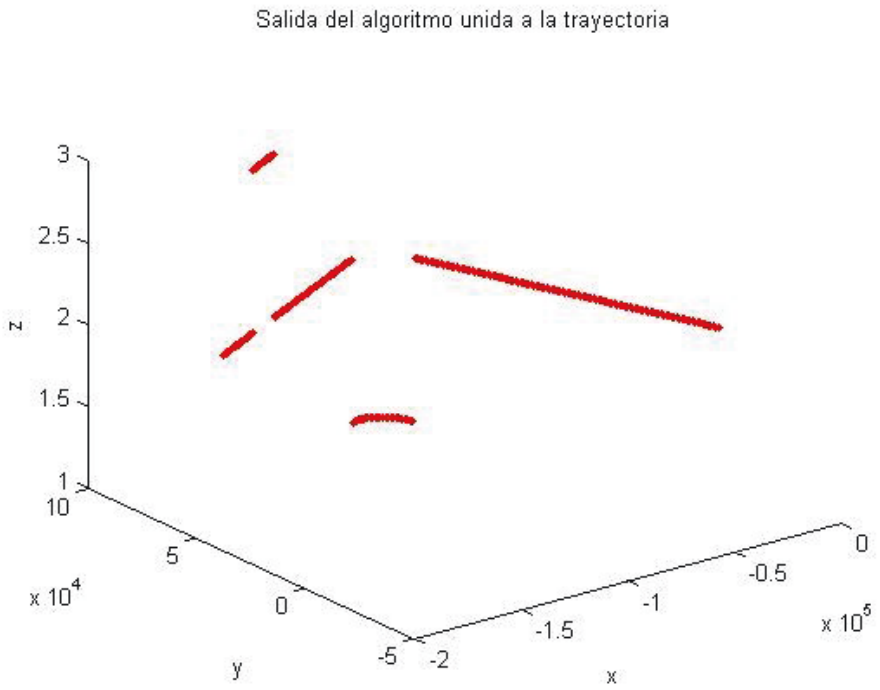


Figura 22B

9.4. Datos con 1 de resolución.

Cuando los datos ya tienen estas resoluciones tan elevadas, mejor se ve el efecto en las curvas, esto se debe a que, como vamos a apreciar en la trayectoria 1, la resolución clasifica de manera regular en las rectas los datos mientras que en las curvas la velocidad disminuye, por lo que los datos se encuentran más próximos, lo que ofrece mayores problemas cuanto más alta sea la resolución.

Trayectoria 1

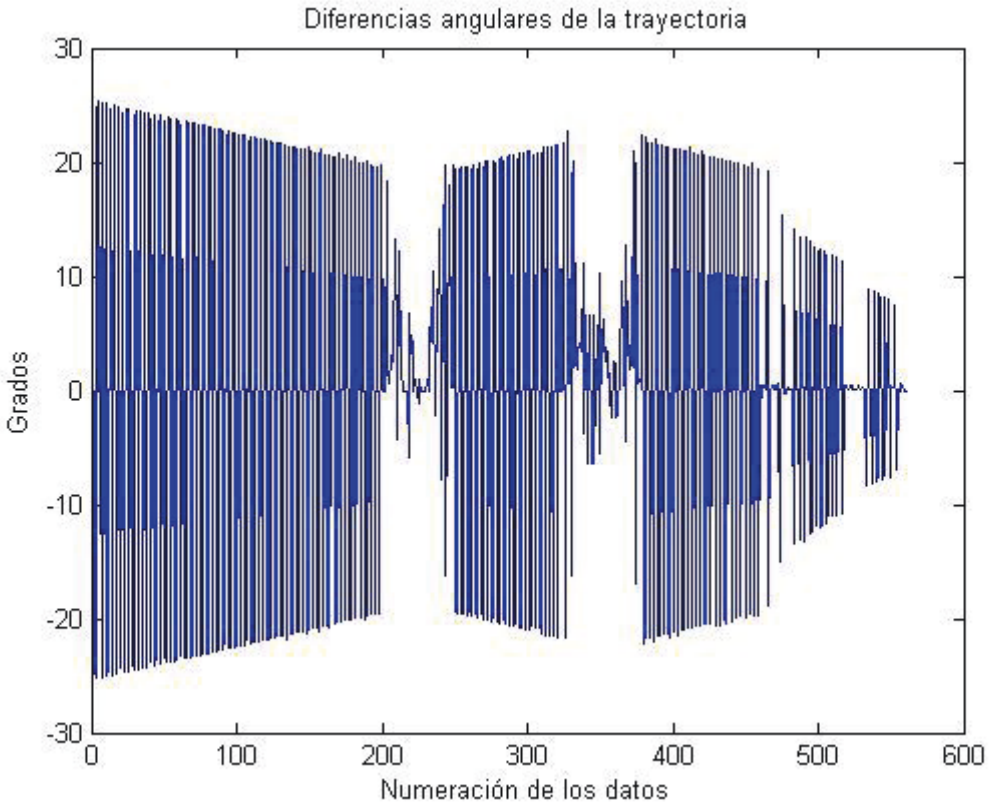


Figura 23A

Es algo con lo que el algoritmo le cuesta trabajar, ya que el modelo está diseñado para que cuando en el observable la diferencia es pequeña, la matriz de observación dé un porcentaje elevado de probabilidad de que sea un estado oculto de recto, pero en este caso eso, no se cumple. Eso hace prever que la discretización de los datos no será fácil. No obstante, veremos que no realiza una clasificación tan mala como se esperaba.

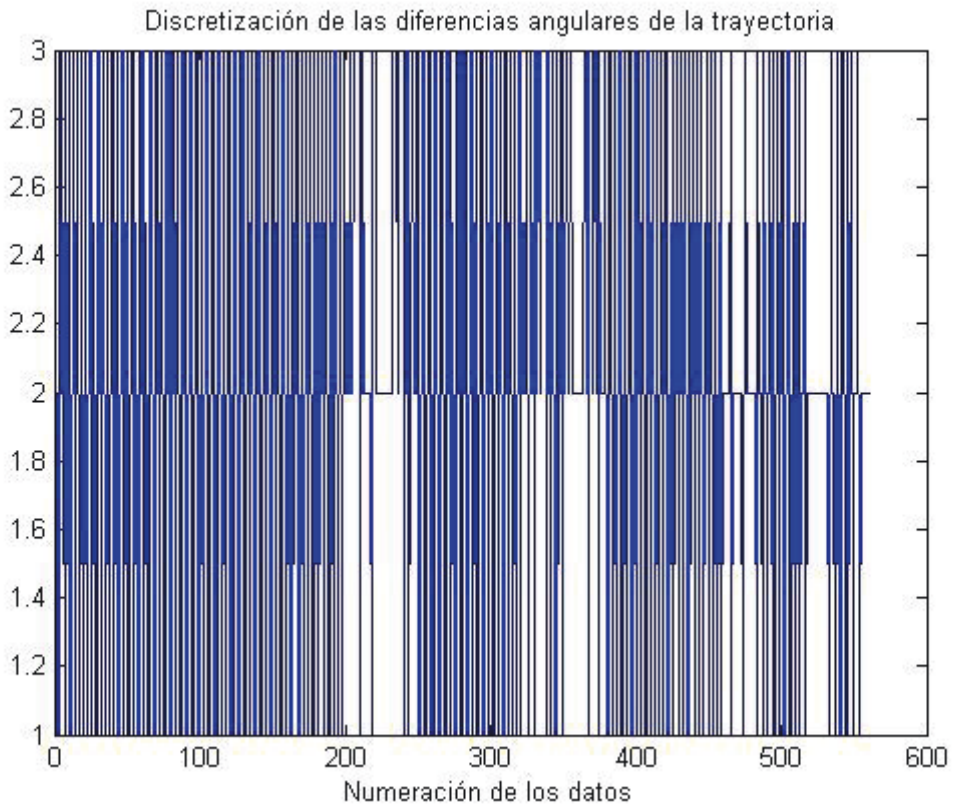


Figura 23B

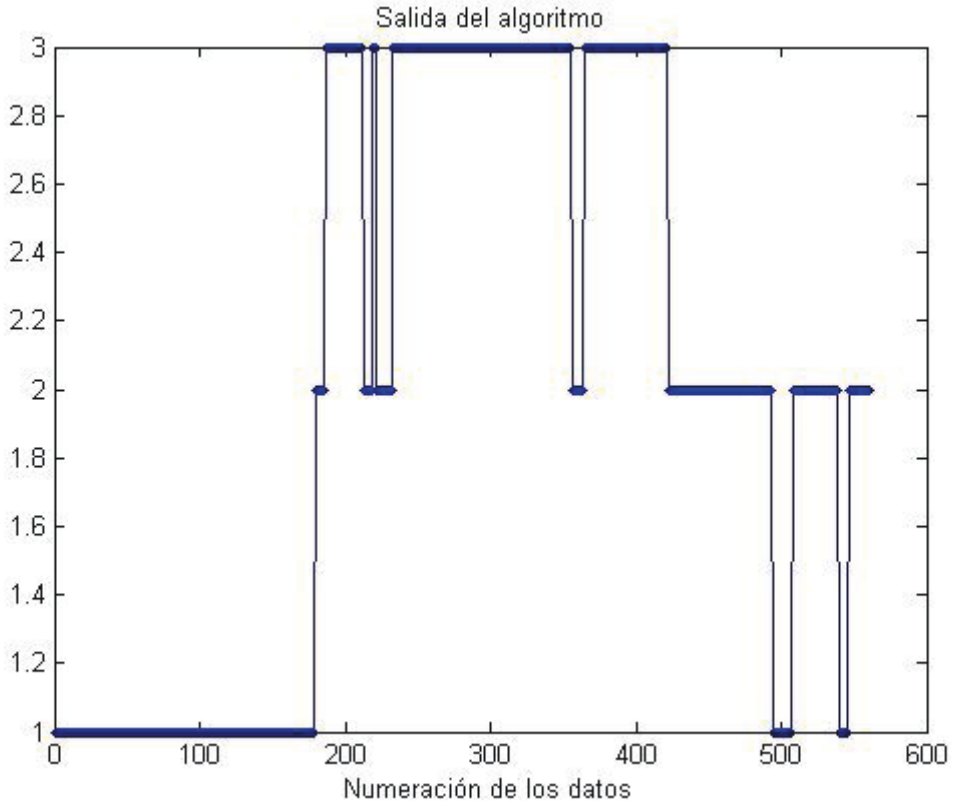


Figura 23C

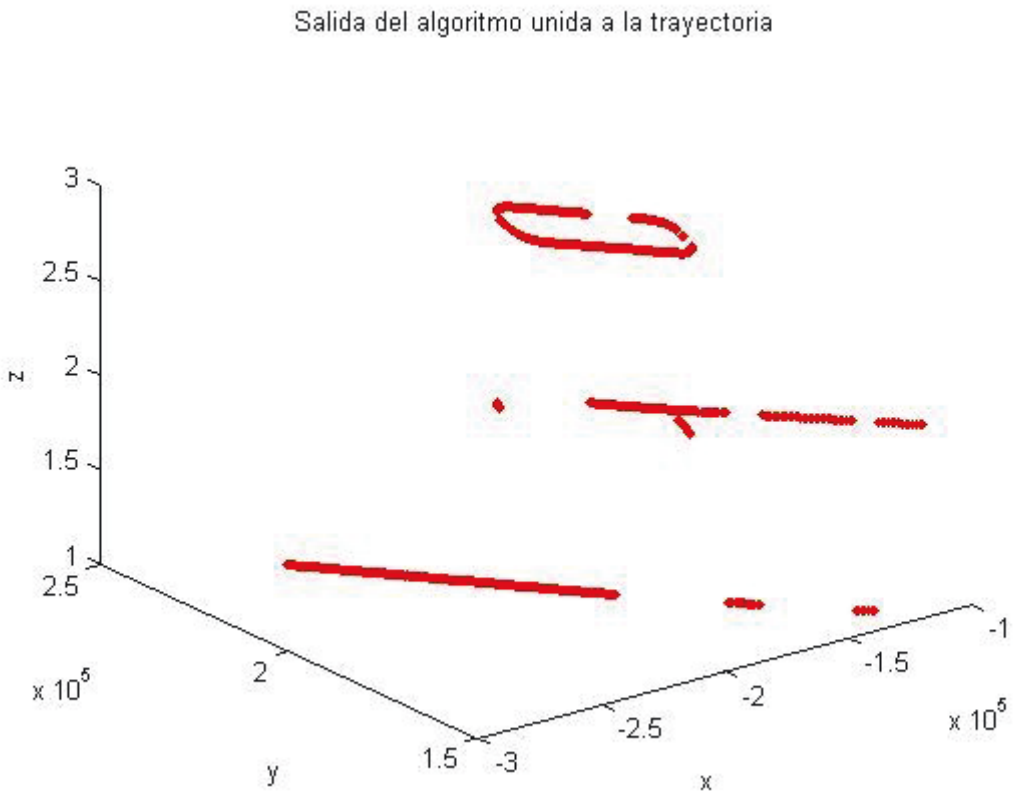


Figura 23D

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,357142857142857	0,253571428571429	0,225000000000000
G. Dcha. (3)	0	0,00892857142857143	0,150000000000000

Tabla 39: Matriz de confusión de la salida con 1 de resolución para trayectoria 1

Observando la matriz de confusión, vemos que clasifica correctamente un 40% de los datos. Aunque los fallos sean muy elevados, el trabajo del algoritmo mejora los datos que nos proporcionaban los observables en gran medida. Veremos cómo la siguientes trayectorias las clasifica algo mejor ya que son más sencillas.

Trayectoria 2

Sigue existiendo el problema de la distancia al sensor, lo que provoca fallos en el algoritmo.

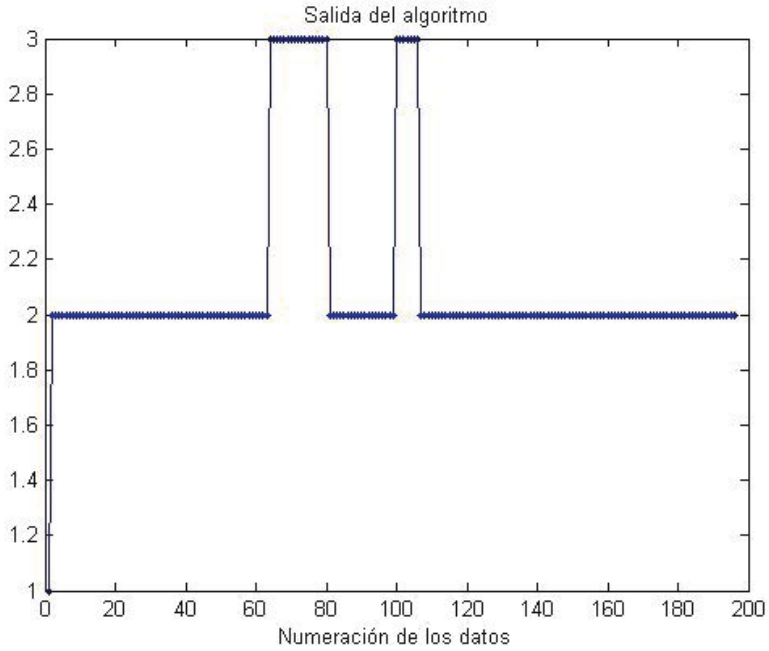


Figura 24

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0,00510204081632653	0,872448979591837	0,122448979591837
G. Dcha. (3)	0	0	0

Tabla 40: Matriz de confusión de la salida con 1 de resolución para trayectoria 2

Al ser una trayectoria más sencilla, clasifica correctamente un 87% de los datos.

Trayectoria 3

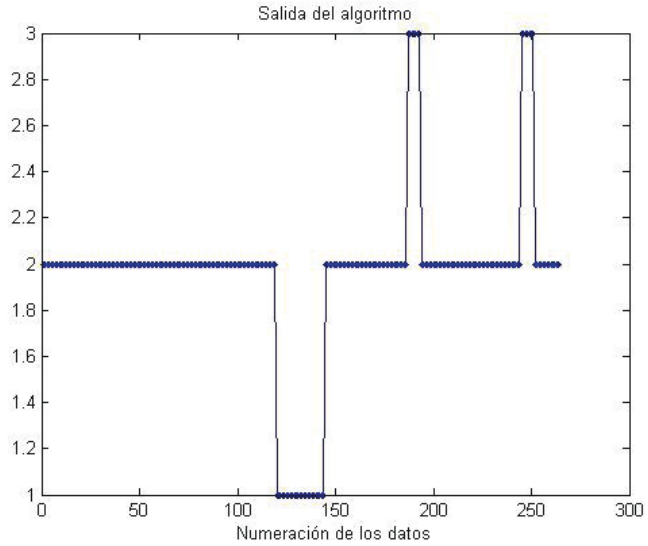


Figura 25A

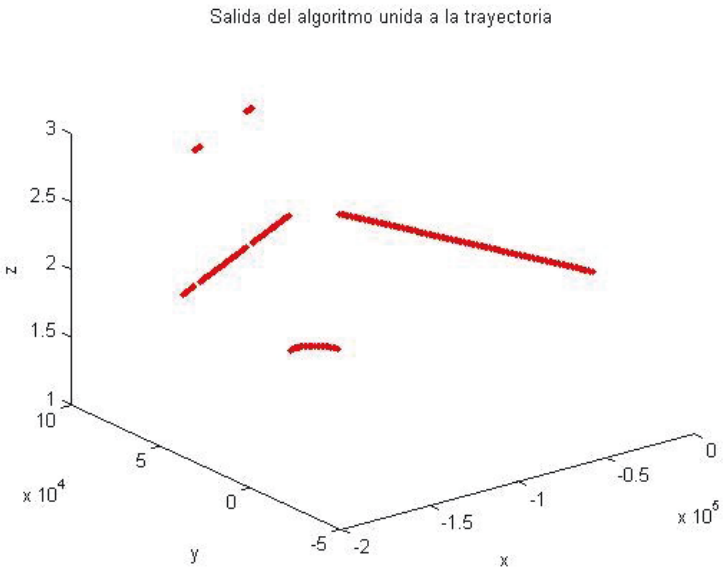


Figura 25B

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0946969696969697	0,00757575757575758	0
Recto (2)	0	0,844696969696970	0,0530303030303030
G. Dcha. (3)	0	0	0

Tabla 41: Matriz de confusión de la salida con 1 de resolución para trayectoria 3

Esta trayectoria ha sido mejor clasificada por el algoritmo. No obstante, aun sigue teniendo fallos, lo más significativo son el 5% de los datos que clasifica como giro a derechas sin tenerlos.

9.5. Datos con 2 de resolución.

Trayectoria 1

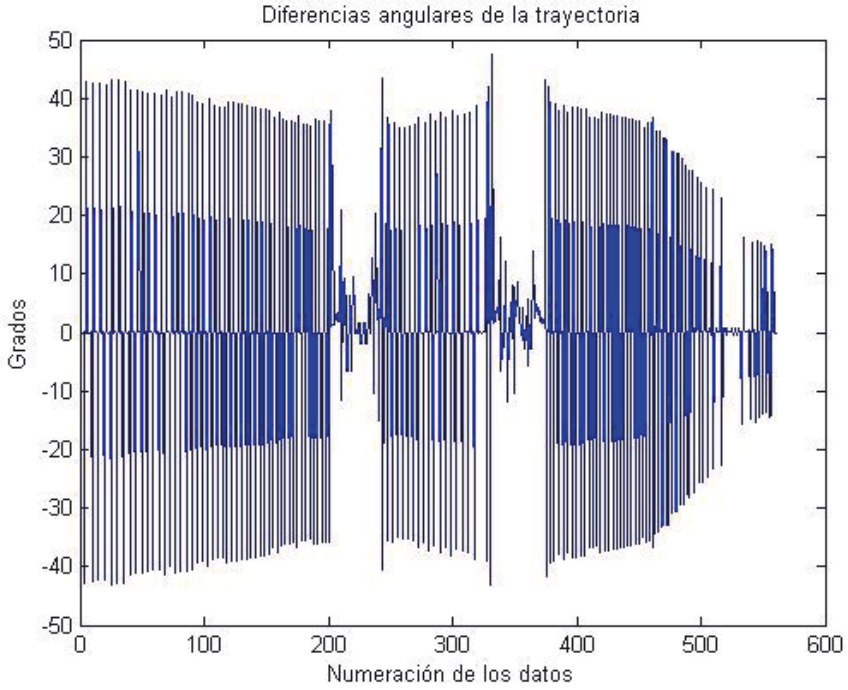


Figura 26A

Tenemos el mismo problema con la resolución y con la distancia al sensor. No obstante, vamos a ver como el algoritmo mejora increíblemente el resultado de la discretización.

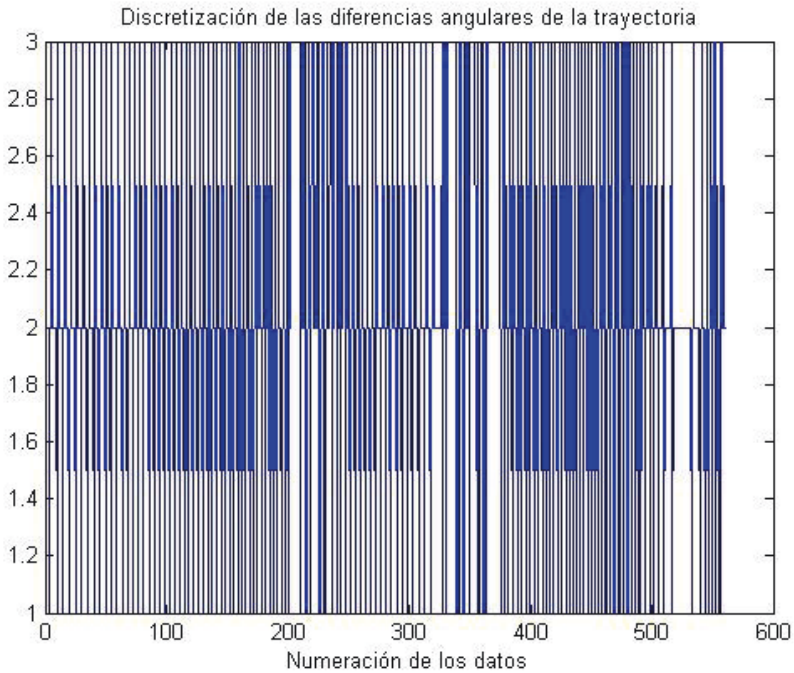


Figura 26B

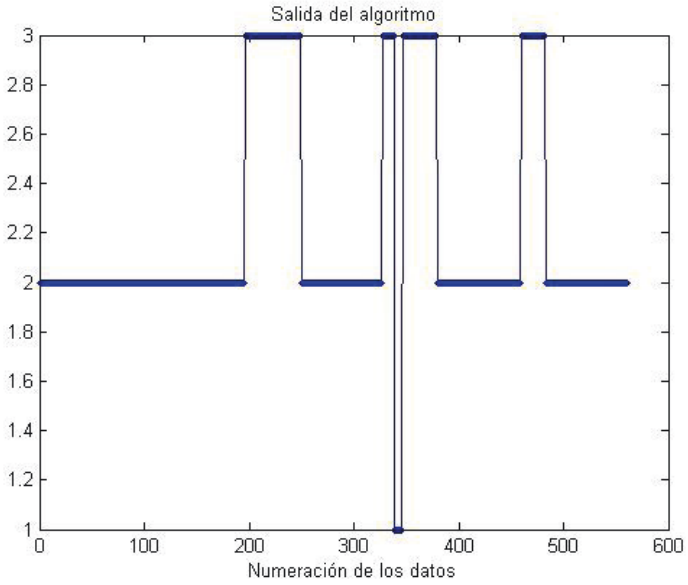


Figura 26C

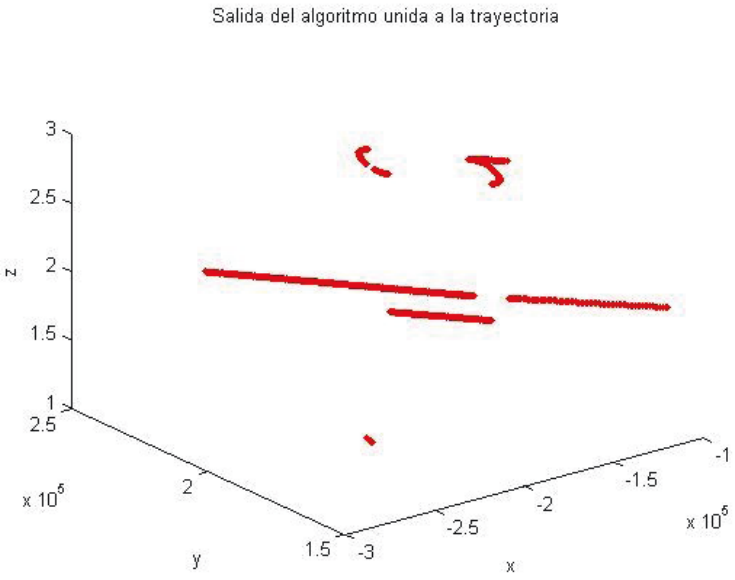


Figura 26D

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0,00535714285714286
Recto (2)	0,0107142857142857	0,748214285714286	0,0767857142857143
G. Dcha. (3)	0,00357142857142857	0,0196428571428571	0,135714285714286

Tabla 42: Matriz de confusión de la salida con 2 de resolución para trayectoria 1

En este caso, doblando la resolución hemos obtenido un 88% de aciertos en la salida del algoritmo, mientras que anteriormente con la trayectoria 1 solo conseguía un 40%. No estamos seguros a qué se debe este cambio, pero demuestra que a pesar de la elevada resolución, puede funcionar correctamente en algunas situaciones.

Trayectoria 2

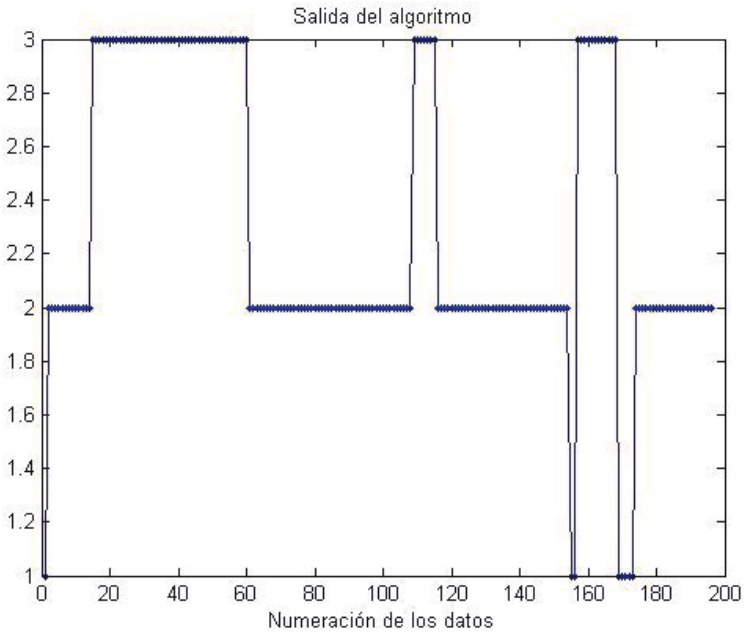


Figura 27

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0,0408163265306122	0,627551020408163	0,331632653061224
G. Dcha. (3)	0	0	0

Tabla 43: Matriz de confusión de la salida con 2 de resolución para trayectoria 2

En la segunda trayectoria, tenemos un resultado menos sorprendente, ya que la resolución y la distancia al sensor crea problemas al clasificar los datos, y por lo tanto empeora el resultado.

Trayectoria 3

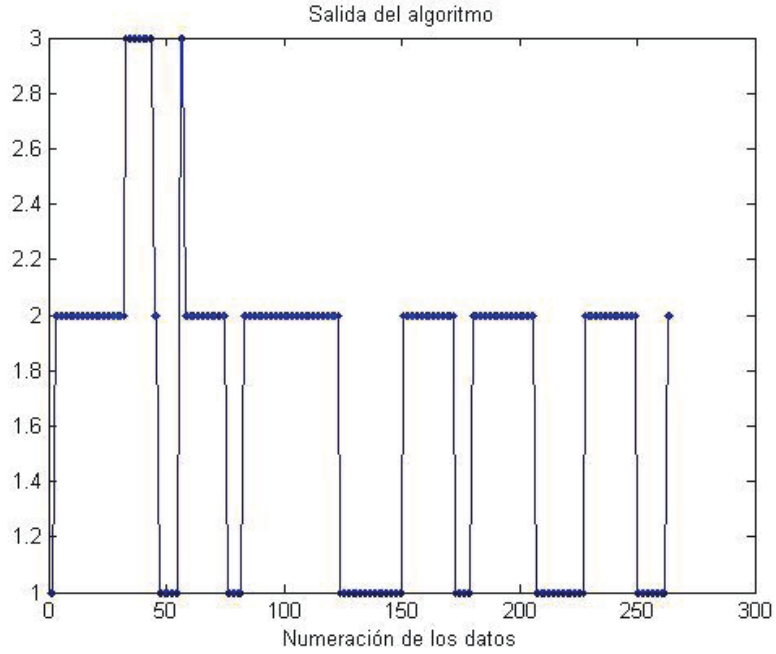


Figura 28A

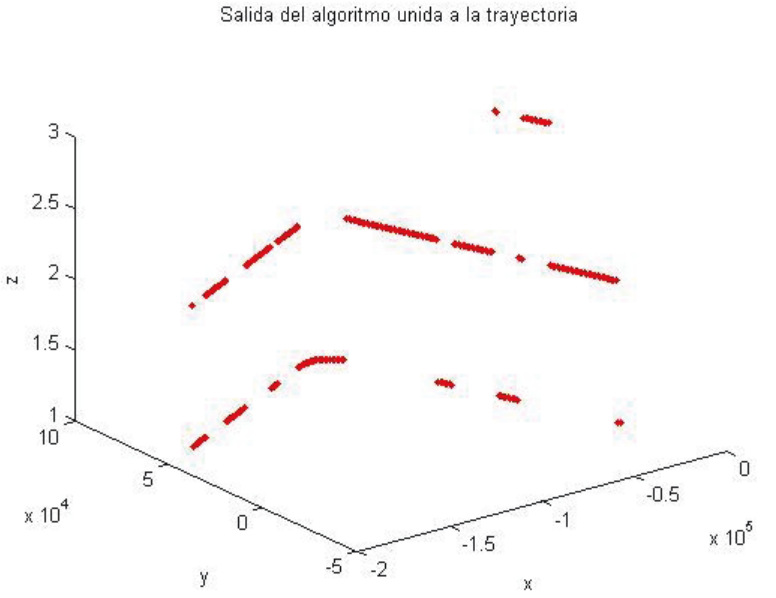


Figura 28B

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0833333333333333	0,0189393939393939	0
Recto (2)	0,242424242424242	0,602272727272727	0,0530303030303030
G. Dcha. (3)	0	0	0

Tabla 44: Matriz de confusión de la salida con 2 de resolución para trayectoria 3

El porcentaje de aciertos es de un 68%, lo que era de esperar para el nivel de resolución en los observables.

9.6. Comparación de aciertos con distintos niveles de resolución:

Trayectorias / Porcentaje de resolución	1	2	3	Media
1/8	99.6%	100%	98.2%	**99.26%**
1/4	91%	89.8%	89.8%	**90.2%**
1/2	92.1%	84.7%	83.1%	**86.63%**
1	40.3%	87.2%	93.9%	**73.8%**
2	88.4%	62.7%	68.5%	**73.2%**

Tabla 45: Porcentaje de aciertos del Viterbi con distintos grados de resolución

Como era de esperar, aunque para algunas trayectorias se consiga un mejor resultado con mayor resolución, al hacer una media entre las tres trayectorias el porcentaje de aciertos disminuye a medida que aumenta el tamaño de las celdas.

10. Modelo con datos simulados con ruido y sin resolución

Ahora quitaremos la resolución y en vez de eso introduciremos ruido en los datos para ver cómo afecta al modelo y a su salida.

Iremos probando con distintos grados de ruido, y para cada uno de ellos iremos modificando el modelo para que se ajuste a los distintos grados de ruido. Vamos a multiplicar el valor nominal de ruido en el simulador por los mismos valores que usamos con resolución (1/8, 1/4, 1/2, 1 y 2). El ruido tiene un efecto más perjudicial para la salida del algoritmo que la resolución como iremos viendo, llegando a un momento en el que el algoritmo no puede trabajar con esa cantidad de ruido.

10.1. Datos con 1/8 de ruido.

Trayectoria 1

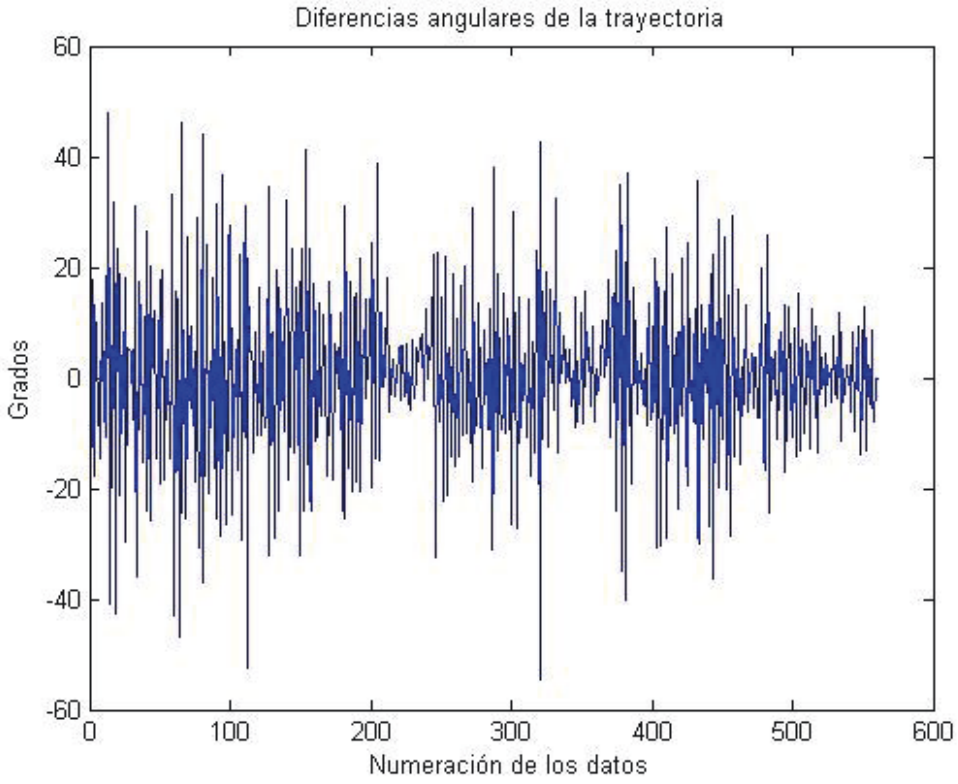


Figura 29A

Como podemos ver en los las diferencias angulares de la trayectoria, no se observan muchos patrones que puedan indicar un giro, y las diferencias angulares son enormes y muy abruptas.

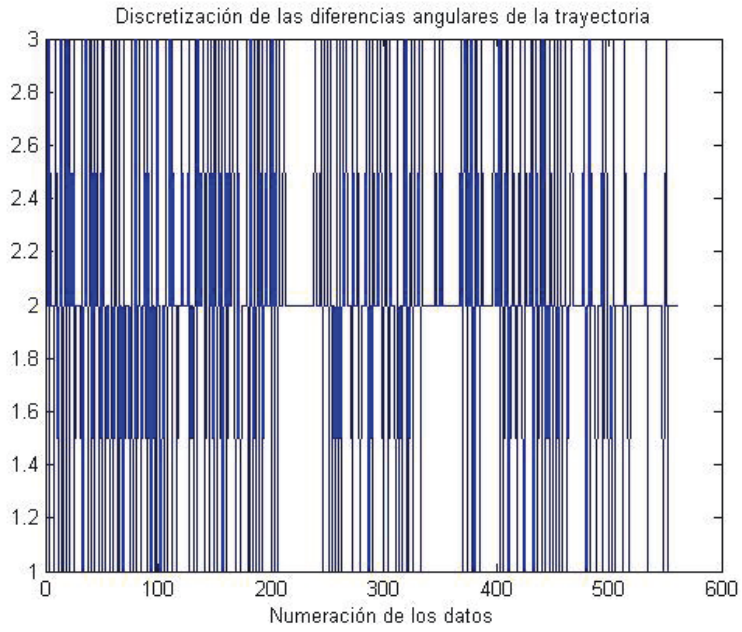


Figura 29B

Una vez discretizamos los datos vemos algún patrón, pero no es muy normal ya que las curvas de la trayectorias coinciden con la clasificación en 2 que se usa para las rectas. No obstante, con una serie de cambios en el modelo se han podido obtener unos resultados mejores de los esperados.

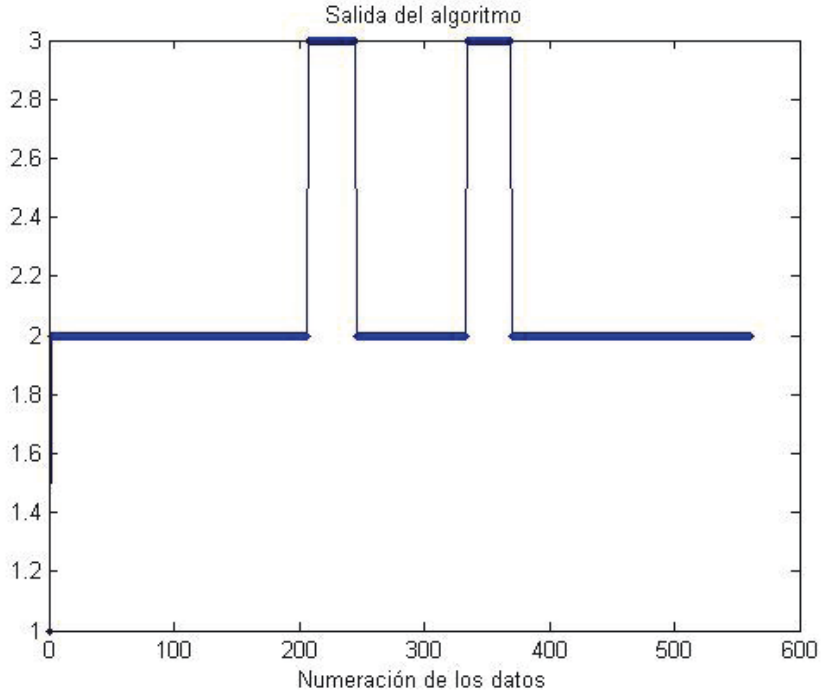


Figura 29C

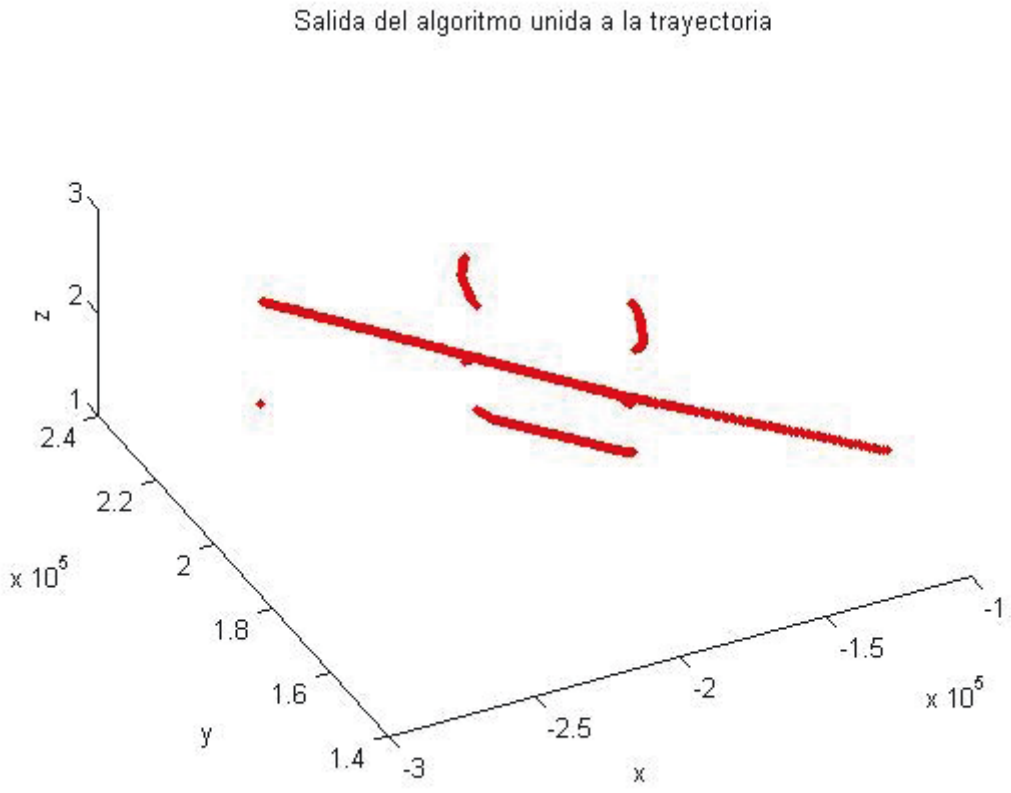


Figura 29D

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00178571428571429	0,782142857142857	0,0517857142857143
G. Dcha. (3)	0	0,0767857142857143	0,0821428571428571

Tabla 46: Matriz de confusión de la salida con 1/8 de ruido para trayectoria 1

Vemos como obtiene las curvas y rectas principales, aunque pierde los detalles. Esto sucede con 1/8 de ruido, lo que nos ayuda a suponer que no será viable aumentar más el ruido.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0,311224489795918	0	0,688775510204082
G. Dcha. (3)	0	0	0

Tabla 47: Matriz de confusión de la salida con 1/8 de ruido para trayectoria 2

La trayectoria 2 la clasifica mal completamente, esto se debe a que hemos debido hacer unos cambios drásticos en el modelo ya que la naturaleza de los datos no era la esperada y con estos cambios hemos mejorado la salida del algoritmo para la trayectoria 1 pero empeorado el resto, y aunque hemos intentado encontrar un punto medio, no se ha conseguido.

Trayectoria 3

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,102272727272727	0	0
Recto (2)	0,715909090909091	0	0,181818181818182
G. Dcha. (3)	0	0	0

Tabla 48: Matriz de confusión de la salida con 1/8 de ruido para trayectoria 2

En la tercera trayectoria se ha conseguido un pequeño avance a la hora de clasificar la curva, pero la parte restante en recta no la clasifica bien. Concretamente se debe a que hemos bajado la probabilidad de observación de las rectas, ya que sino en la trayectoria 1 nos clasificaba toda la trayectoria como una sola recta.

10.2. Datos con 1/4 de ruido.

Trayectoria 1

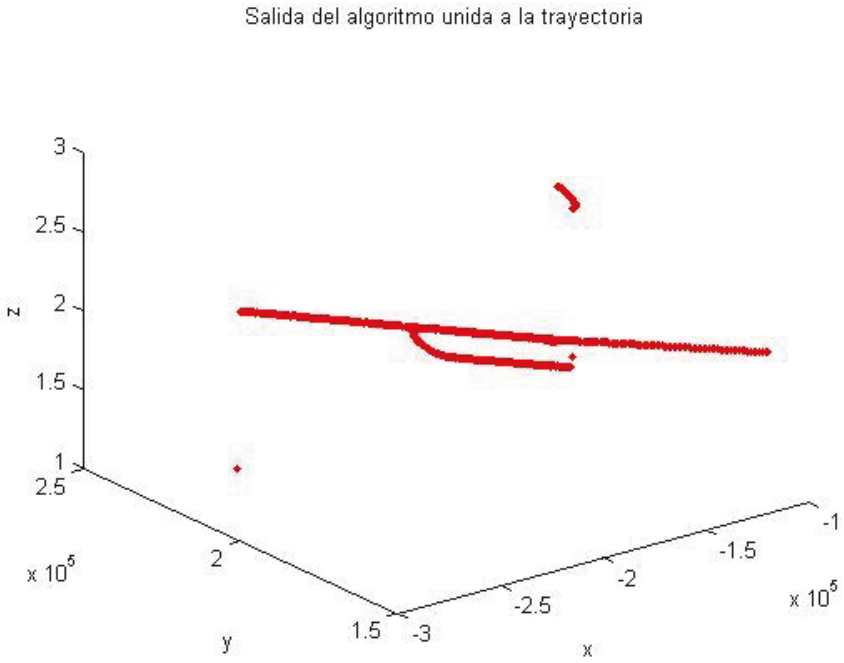


Figura 30

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00178571428571429	0,812500000000000	0,0214285714285714
G. Dcha. (3)	0	0,114285714285714	0,0446428571428571

Tabla 49: Matriz de confusión de la salida con 1/4 de ruido para trayectoria 1

Como se puede ver, consigue clasificar una parte bien, pero los resultados son negativos. Se ha perdido un gran porcentaje de los giros.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0,698979591836735	0,158163265306122	0,142857142857143
G. Dcha. (3)	0	0	0

Tabla 50: Matriz de confusión de la salida con 1/4 de ruido para trayectoria 2

Trayectoria 3

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,102272727272727	0	0
Recto (2)	0,791666666666667	0,0265151515151515	0,0795454545454545
G. Dcha. (3)	0	0	0

Tabla 51: Matriz de confusión de la salida con 1/4 de ruido para trayectoria 3

Los resultados de las trayectorias 2 y 3 tienen un porcentaje de acierto muy bajo, y ya que el ruido va a ir creciendo, éste no va a mejorar mucho.

Si aumentamos más el ruido, los porcentajes de aciertos no son concluyentes, ya que la mayoría de las ocasiones el algoritmo clasifica todo a la opción de recto, por incapacidad de reconocer las transiciones en los observables, ya que estos tienen demasiado ruido, y si consigue un buen porcentaje de aciertos será porque en las trayectorias predominan las rectas.

10.3. Comparación de aciertos con distintos niveles de ruido:

Aunque no mostremos los resultados de aumentar el ruido, si tenemos los porcentajes de aciertos, por lo que vamos a comprarlos.

Trayectorias / Porcentaje de ruido	1	2	3	Media
1/8	86.6%	0%	10.2%	**32.26%**
1/4	85.7%	15.8%	12.8%	**38.1%**
1/2	73.7%	100%	89.7%	**87.8%**
1	83.3%	100%	88.2%	**90.5%**
2	83.3%	100%	89%	**90.7%**

Tabla 52: Porcentaje de aciertos del Viterbi con distintos grados de ruido

Vemos que los porcentajes son altos por lo mencionado anteriormente, pero no tienen ninguna validez ya que en trayectorias con más curvas los resultados serían totalmente opuestos.

11. Modelo con datos simulados sin ruido, con resolución y filtro de Kalman

El filtro de Kalman se encarga de disminuir el efecto del ruido y de la resolución en las trayectorias, pero tiene un problema, cuando se aumenta el ruido o la resolución, éste lo disminuye, pero también añade un retraso en las diferencias angulares cuando se produce un giro en la trayectoria, por lo que los datos que nos llegan tienen menos errores pero la curva la observamos ligeramente más tarde de cuando en realidad se lleva a cabo. Esto dificultará el funcionamiento del algoritmo por el retraso, pero mejora los problemas causados por el ruido o la resolución.

En este caso los vectores A y B están formados por las velocidades en un punto normalizadas en los ejes X e Y, estas velocidades han sido obtenidas mediante el filtro de Kalman. Pero la entrada del modelo seguirán siendo las mismas, las diferencias angulares discretizadas, sólo que esta vez las obtendremos de la velocidad, y la anterior de la diferencia entre las posiciones.

```
for i=1:(length(A)-1),
 x = A(i)
 y = B(i);
 angulo =atan2(y,x)*180/pi;
 C(i)= angulo;
 if (i>1)
    dif=C(i-1)-angulo;
    while(abs(dif-360)< abs(dif))
         dif= dif -360;
    end
    while(abs(dif+360)< abs(dif))
         dif= dif +360;
    end
    D(i-1)= dif;
    D(i)=0;
    D(i+1)=0;
 end
end
```

Figura 31: Código de creación de los observables (con filtro de Kalman)

11.1. Datos con 1/8 de resolución.

A los datos con una resolución tan baja no es necesario aplicar el filtro de Kalman. Es posible, incluso, que al aplicarlo, añada un retardo que empeore el resultado. No obstante, veremos que cuando aumentemos la resolución o añadamos ruido a las trayectorias, mejoraremos los resultados.

Como venimos viendo, en función del ruido de los datos o del tamaño de resolución, usaremos unas medidas u otras para realizar la discretización de los datos, al igual que ajustaremos el modelo para obtener una mejor salida del algoritmo con las distintas entradas.

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.65	0.25	0.1
Recto (2)	0.15	0.7	0.15
Giro Dcha. (3)	0.1	0.25	0.65

Tabla 53: Matriz de transición del modelo para datos con 1/8 de resolución y filtro de Kalman

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.6	0.35	0.05
Diferencia angular pequeña (2)	0.1	0.8	0.1
Diferencia angular grande positiva (3)	0.05	0.35	0.65

Tabla 54: Matriz de observación del modelo para datos con 1/8 de resolución y filtro de Kalman

Además de modificar los topes de la discretización de 1 a -1.

Trayectoria 1

Vamos a ver cómo en los siguientes gráficos el efecto del filtro de Kalman reduce las variaciones bruscas entre las diferencias angulares.

Vemos un ejemplo que tenemos con 1/8 de resolución sin aplicar el filtro, y luego el que vamos a estudiar con el filtro.

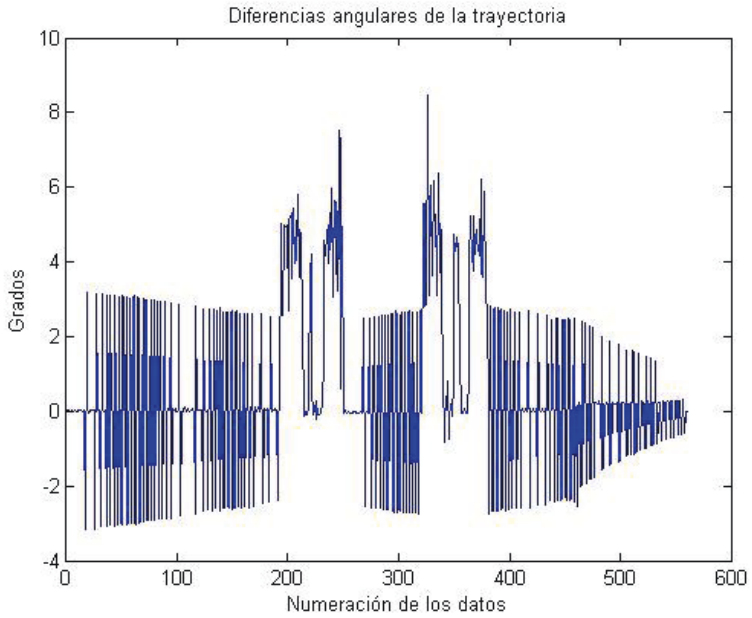


Figura 32A

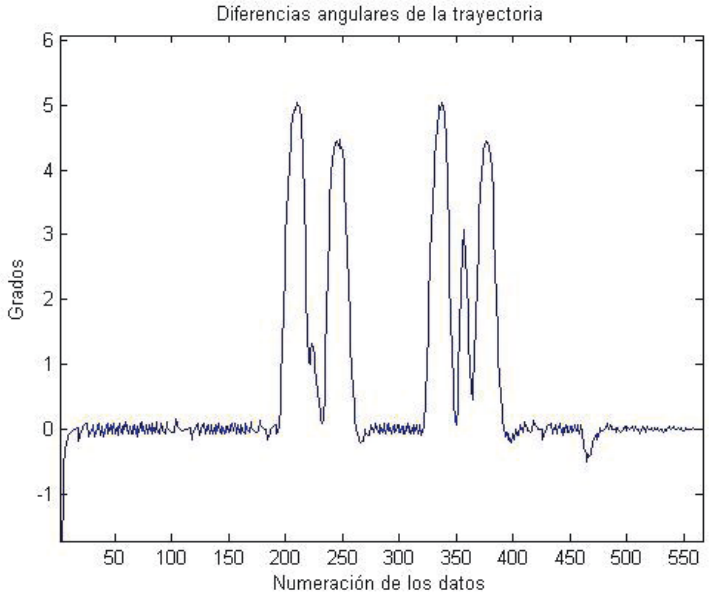


Figura 32B

Las diferencias entre ambos son notables, ya que en las rectas disminuye la diferencia angular a 0.2 aproximadamente, aunque al suavizar la trayectoria vemos como la pequeña curva entre las dos grandes primeras se pierde casi por completo.

La discretización de los datos es prácticamente perfecta, al igual que la salida del algoritmo.

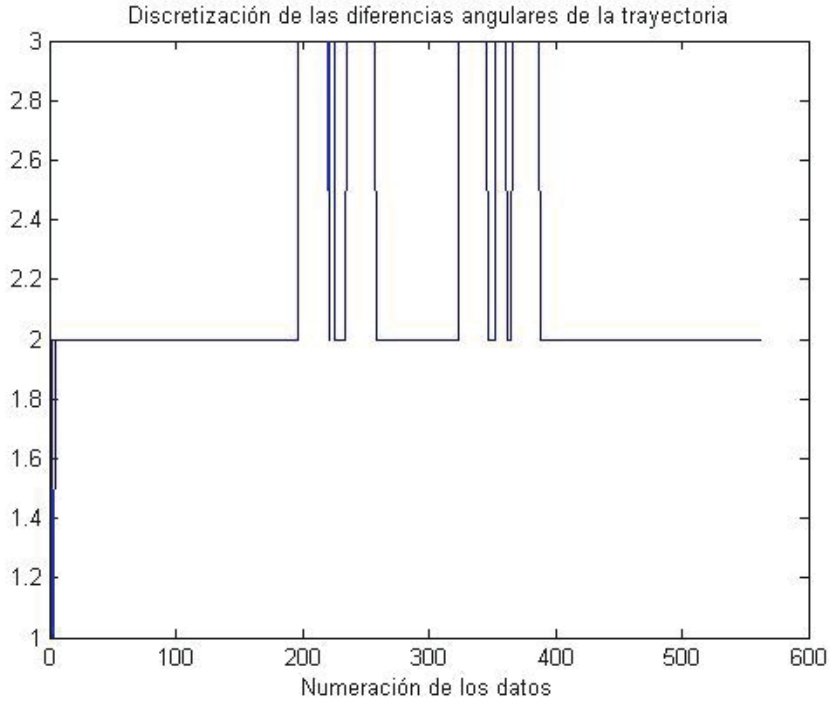


Figura 32C

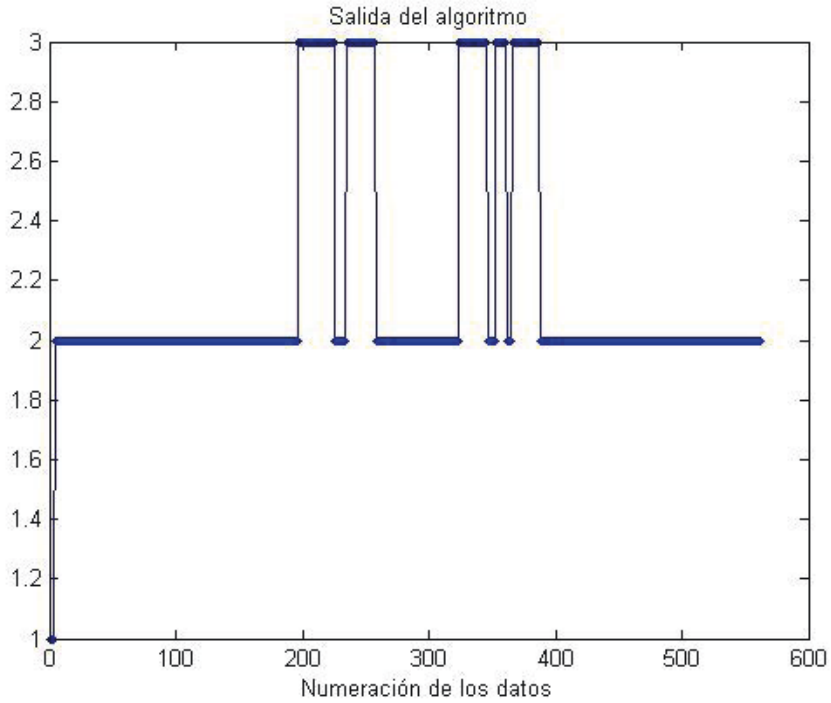


Figura 32D

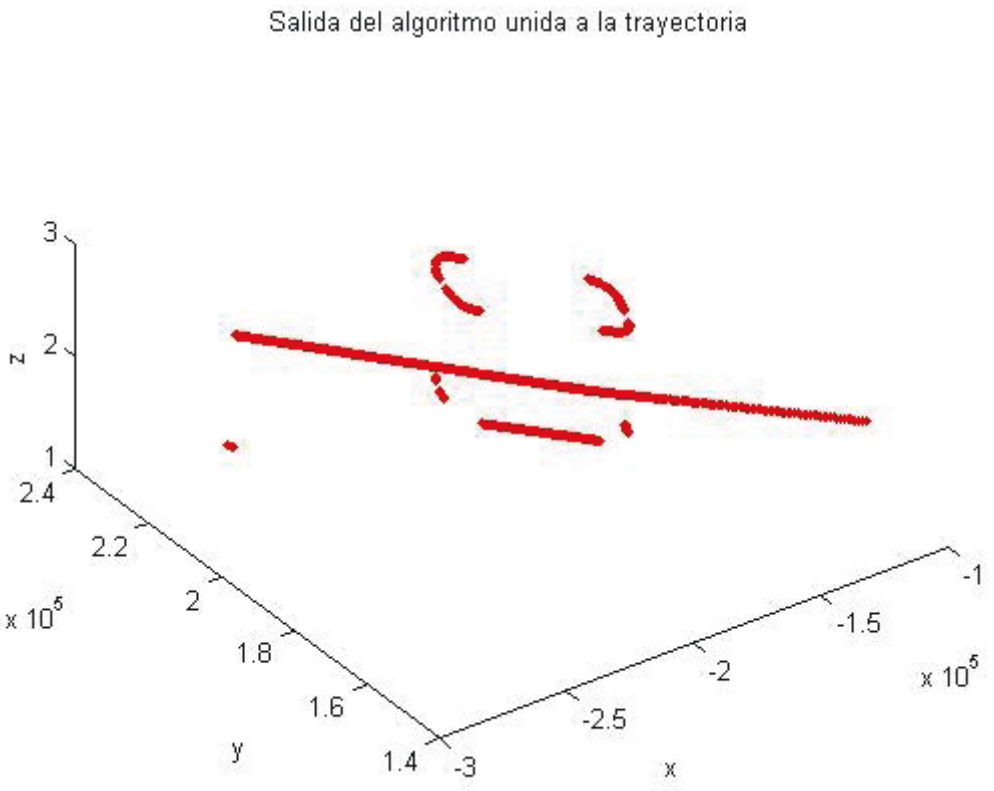


Figura 32E

Usaremos la matriz de confusión para comprobar la salida y sus fallos.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00714285714285714	0,767857142857143	0,0607142857142857
G. Dcha. (3)	0	0,0303571428571429	0,128571428571429

Tabla 55: Matriz de confusión de la salida con 1/8 de resolución y filtro de Kalman para trayectoria 1

Como podemos ver el porcentaje de error está entorno a un 10%.

Trayectoria 2

Recordamos que la trayectoria 2 es una simple recta, por lo que no nos centraremos en exceso en ella, ya que el filtro de Kalman en las rectas funciona con total normalidad.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	0,989795918367347	0,0102040816326531
G. Dcha. (3)	0	0	0

Tabla 56: Matriz de confusión de la salida con 1/8 de resolución y filtro de Kalman para trayectoria 2

Como era de esperar el resultado es prácticamente de un 100% de aciertos.

Trayectoria 3

Es una trayectoria con una sola curva que con este tamaño de resolución no debe de dar muchos problemas. No obstante, veremos los datos que nos proporciona el filtro de Kalman y la salida del algoritmo.

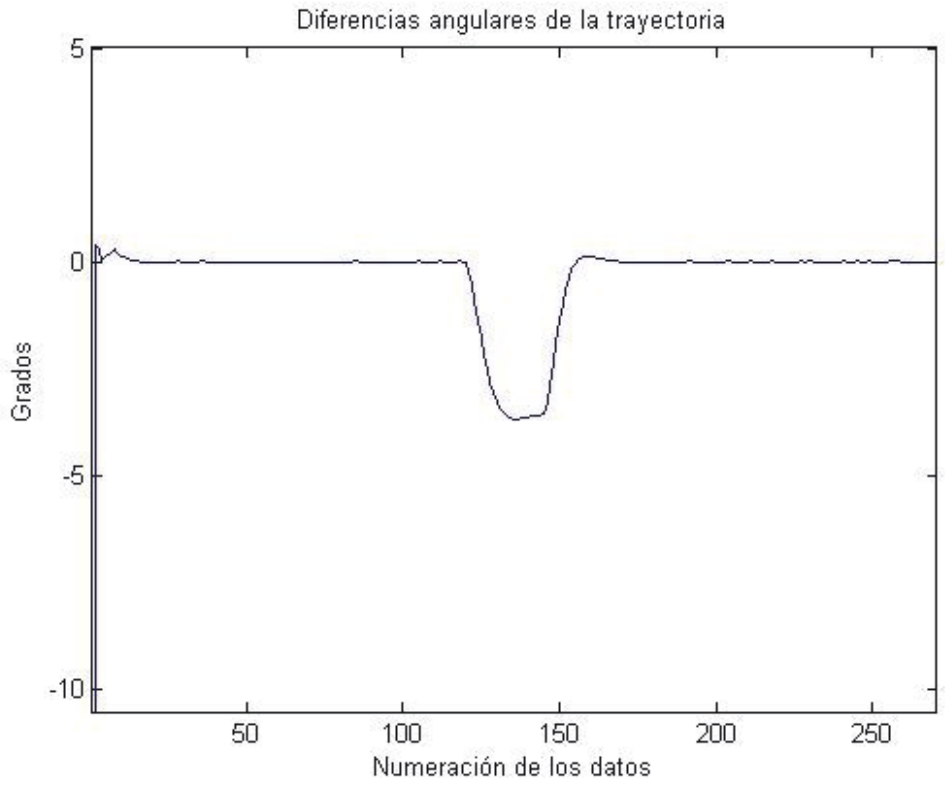


Figura 33A

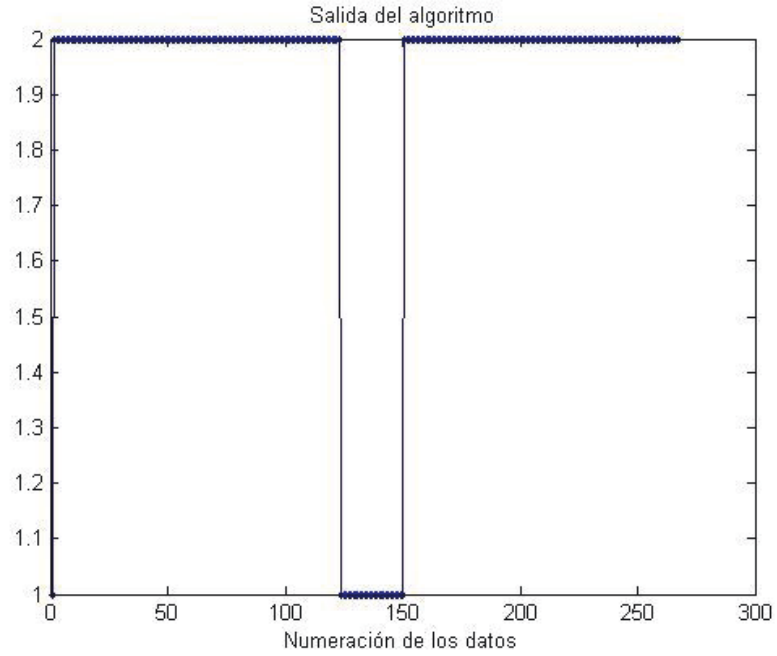


Figura 33A

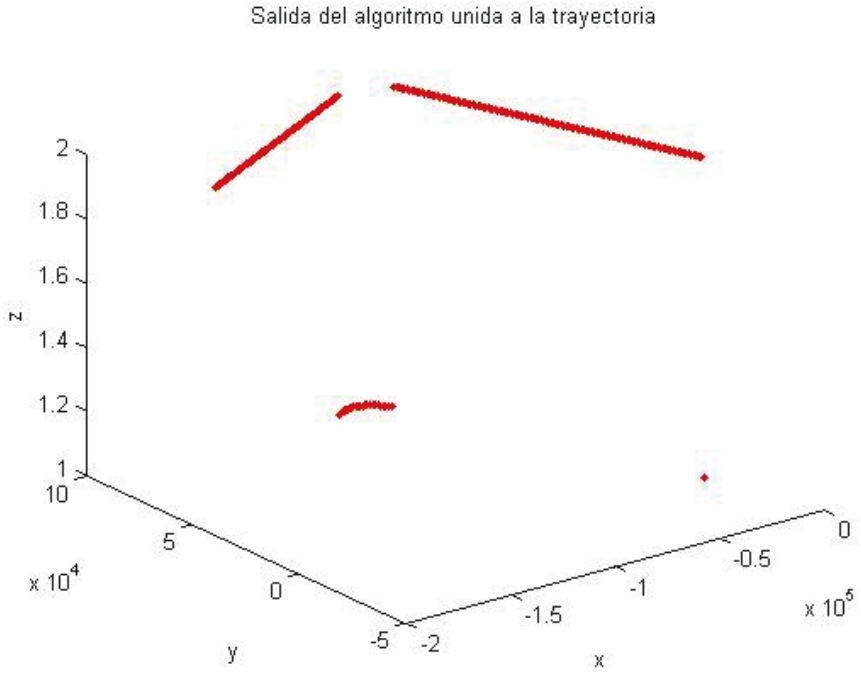


Figura 33B

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0833333333333333	0,0189393939393939	0
Recto (2)	0,0227272727272727	0,875000000000000	0
G. Dcha. (3)	0	0	0

Tabla 57: Matriz de confusión de la salida con 1/8 de resolución y filtro de Kalman para trayectoria 3

Como podemos ver, con esta resolución los errores en la salida del algoritmo son mínimos. Seguramente tengamos más errores en las trayectorias con curvas porque será donde el filtro de Kalman nos proporcione peores datos, y más en concreto en las trayectorias con curvas menos pronunciadas porque seguramente tienda a eliminarlas.

11.2. Datos con 1/4 de resolución.

Para esta resolución, hemos hecho pequeñas variaciones en el modelo y a la hora de la discretización para mejorar la salida del algoritmo.

Trayectoria 1

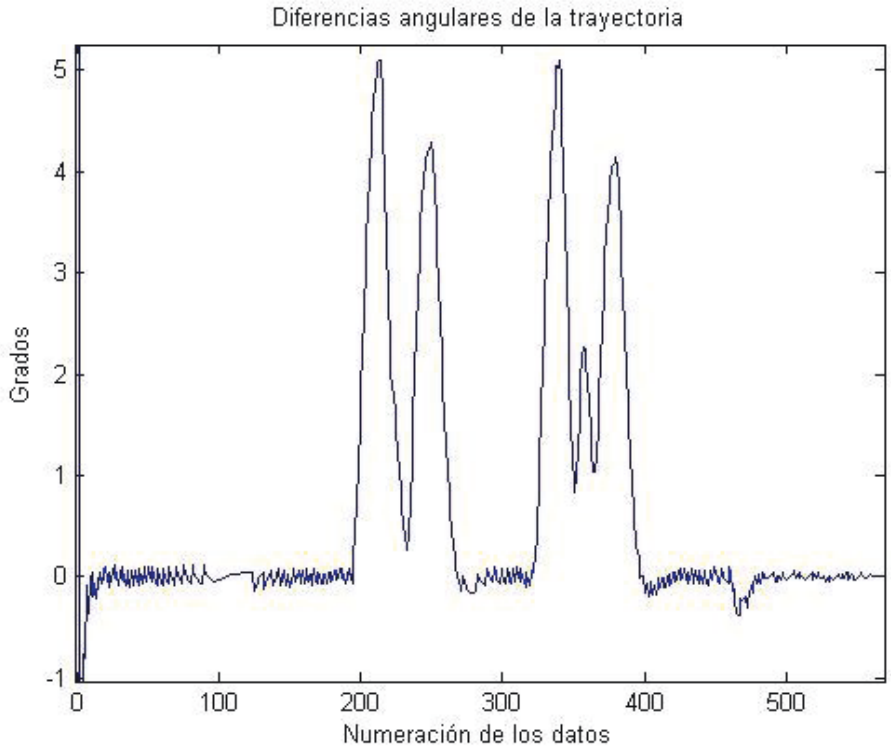


Figura 34A

Como vemos con los datos de entrada, ya ni vemos la curva que se produce después del primer giro lo que nos indica que el algoritmo va a cometer el mismo error, dado que no puede prever curvas en las que no aparece ninguna diferencia angular de ningún tipo.

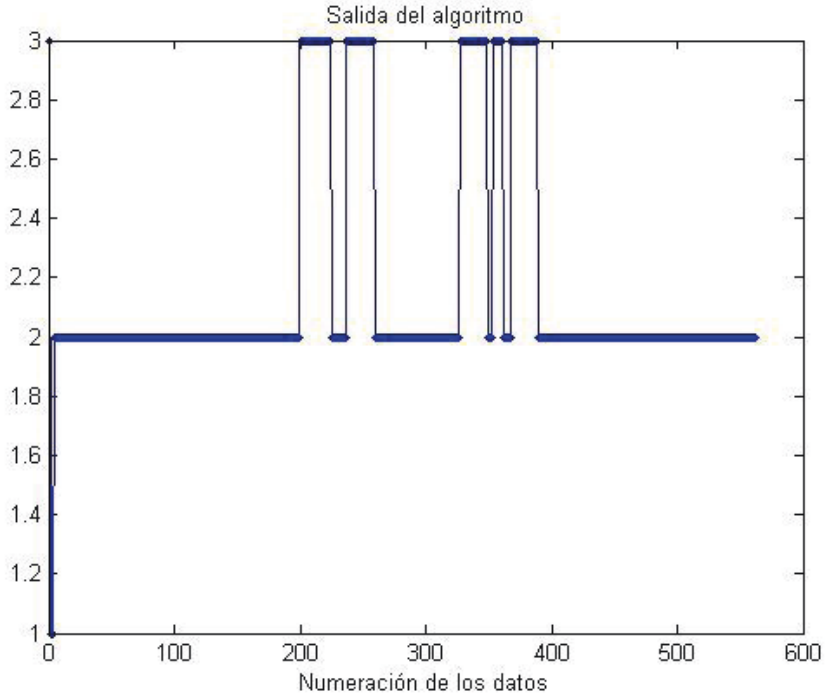


Figura 34B

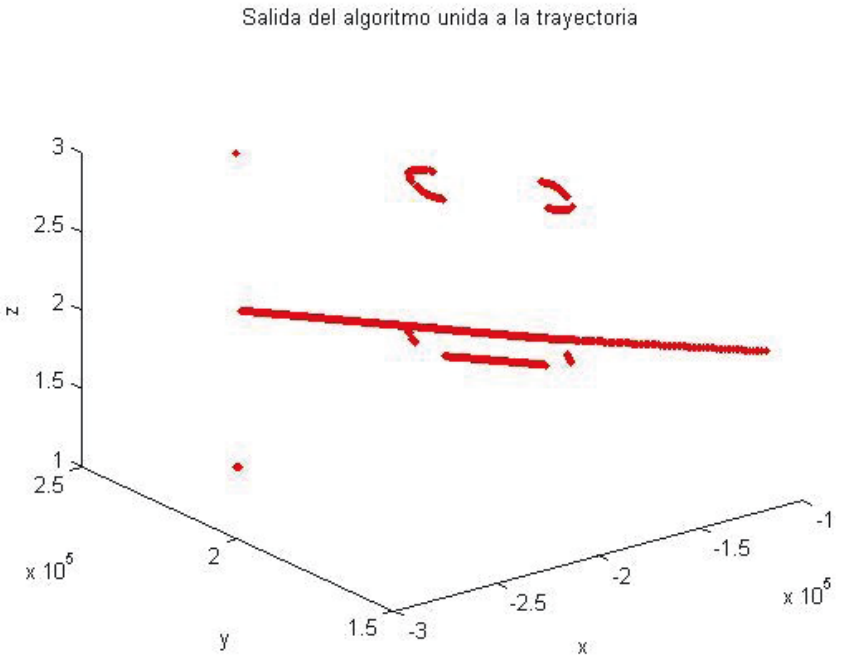


Figura 34C

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00535714285714286	0,760714285714286	0,0696428571428572
G. Dcha. (3)	0	0,0500000000000000	0,108928571428571

Tabla 58: Matriz de confusión de la salida con 1/4 de resolución y filtro de Kalman para trayectoria 1

Como vemos en la matriz, comienzan a verse los errores producidos por el retraso de la curva en los datos proporcionados por el filtro de Kalman.

Prácticamente tenemos un 5% de error clasificando recto por derecha, y otro 5% derecha por recto. Esto se debe a que retrasa la curva, por tanto la inicia y la termina más tarde, provocando que los errores entre las curvas y las rectas aumenten proporcionalmente.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 59: Matriz de confusión de la salida con 1/4 de resolución y filtro de Kalman para trayectoria 2

La clasifica totalmente perfecta, lo que no es de sorprender al ser una recta y teniendo en cuenta que el filtro de Kalman suaviza las variaciones de diferencias angulares producidas por la resolución o el ruido.

Trayectoria 3

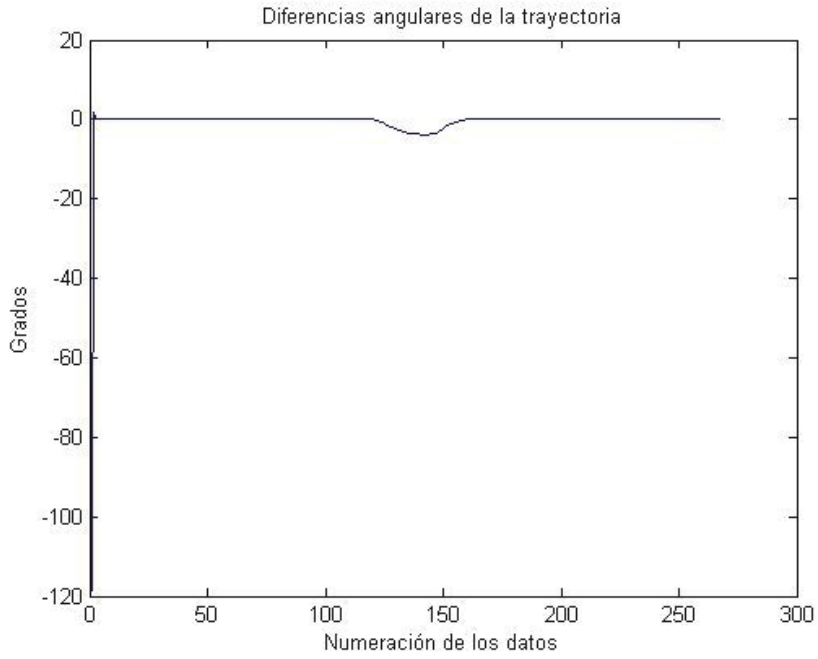


Figura 35A

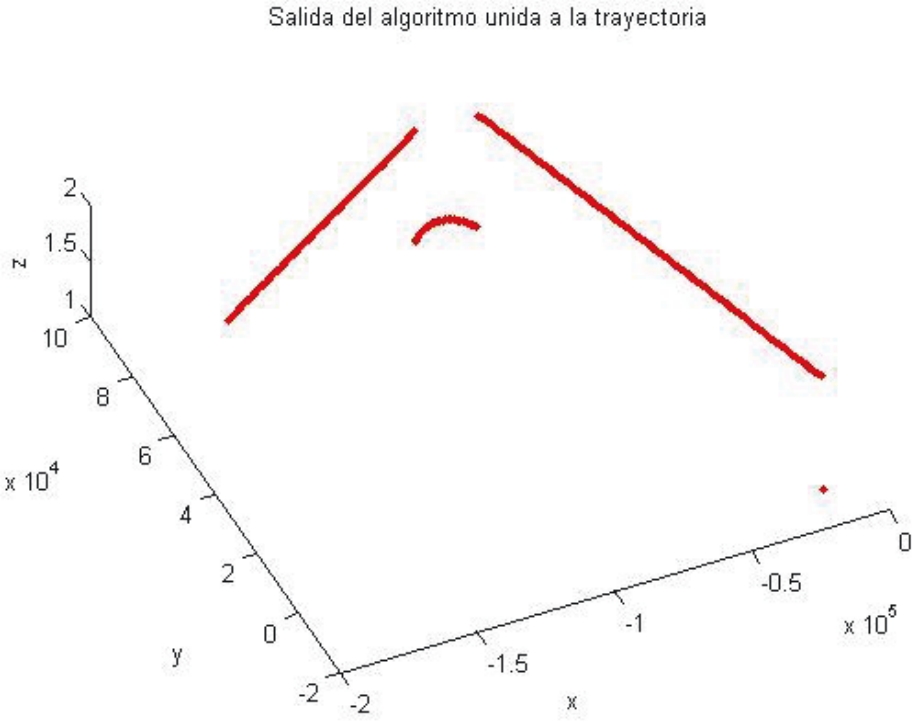


Figura 35B

En los dos anteriores gráficos, hemos podido ver el efecto el retardo en la curva para los datos de entrada y para la salida del algoritmo. Y como hemos mencionado antes, al ser un retardo cabe esperar que los errores de entre recta y giro sean proporcionales, lo que podemos ver en la siguiente matriz de confusión.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00714285714285714	0,751785714285714	0,0767857142857143
G. Dcha. (3)	0	0,0625000000000000	0,0964285714285714

Tabla 60: Matriz de confusión de la salida con 1/4 de resolución y filtro de Kalman para trayectoria 3

11.3. Datos con 1/2 de resolución.

Los cambios en el modelo son mínimos, aun así un ligero retoque es necesario para mejorar levemente el resultado del algoritmo.

Trayectoria 1

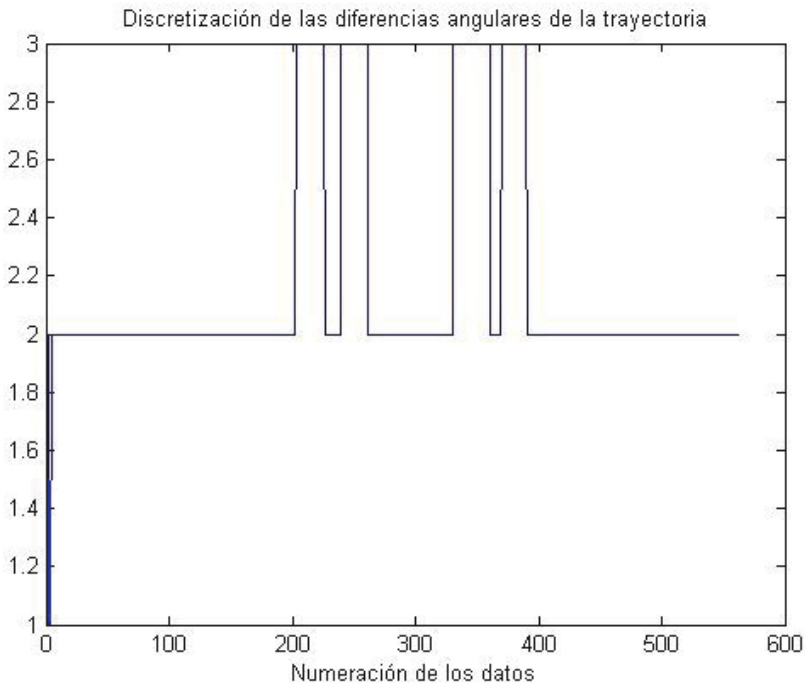


Figura 36A

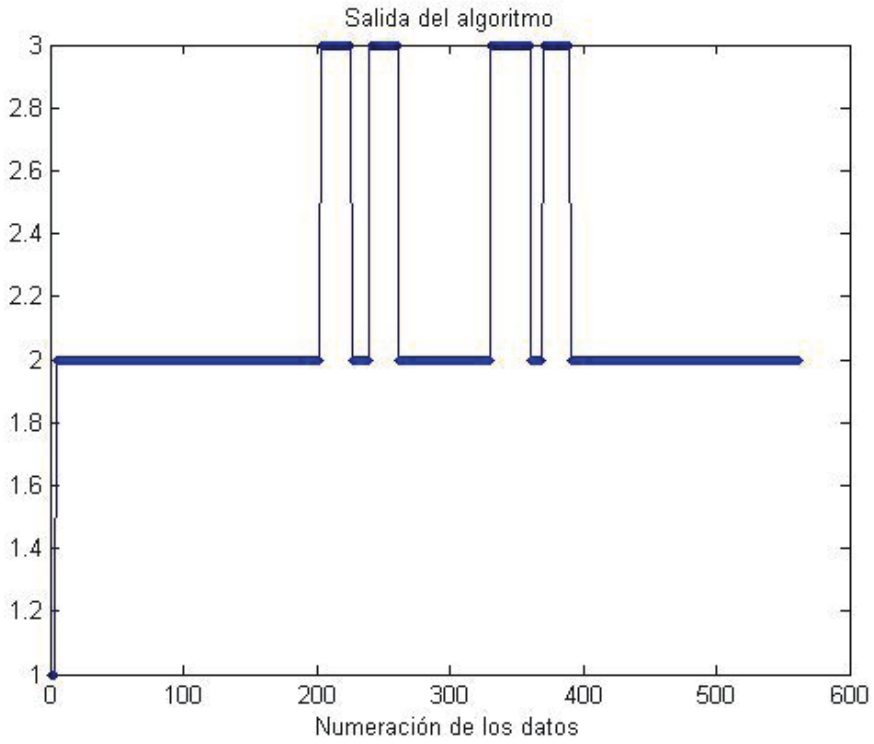


Figura 36B

Vemos como la salida parece correcta en el gráfico superior, mientras que en el gráfico inferior se observa el retraso que tiene en la clasificación de la curva. Como ya hemos mencionado, el problema se debe al filtro de Kalman, que disminuye enormemente el ruido y los efectos de la resolución, pero añade este problema.

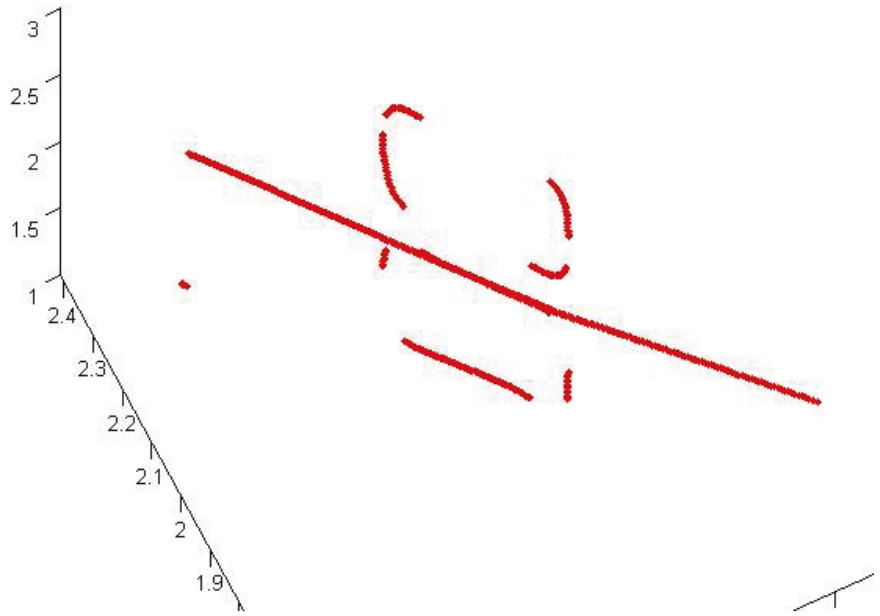


Figura 36C

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00714285714285714	0,751785714285714	0,0767857142857143
G. Dcha. (3)	0	0,0625000000000000	0,0964285714285714

Tabla 61: Matriz de confusión de la salida con 1/2 de resolución y filtro de Kalman para trayectoria 1

En la matriz de confusión vemos cómo se reflejan los errores observados en los gráficos, y el algoritmo junto al modelo no puede corregir el problema que nos añade el filtro de Kalman.

Trayectoria 2

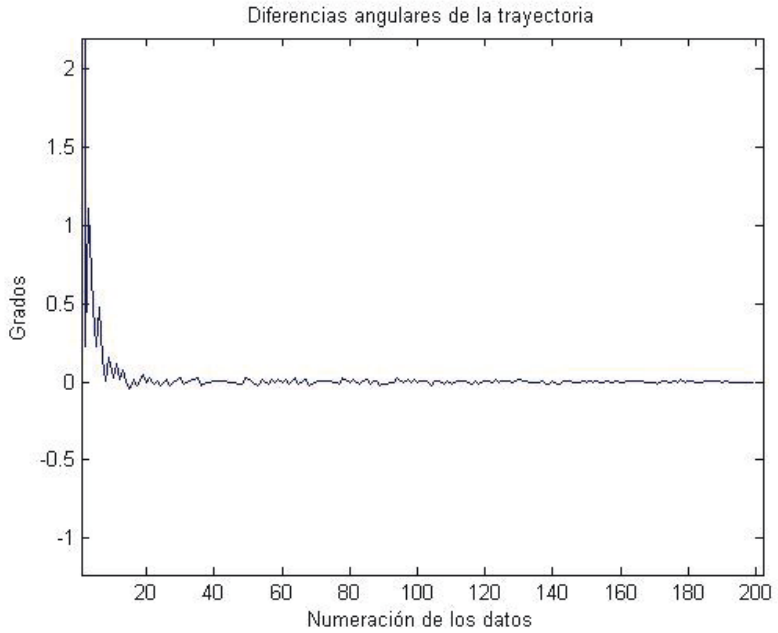


Figura 37

Con la segunda trayectoria, no tiene ningún problema, al no tener curvas el filtro de Kalman funciona a la perfección y reduce el efecto de la resolución completamente, por lo que la salida del algoritmo es perfecta.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 62: Matriz de confusión de la salida con 1/2 de resolución y filtro de Kalman para trayectoria 2

Trayectoria 3

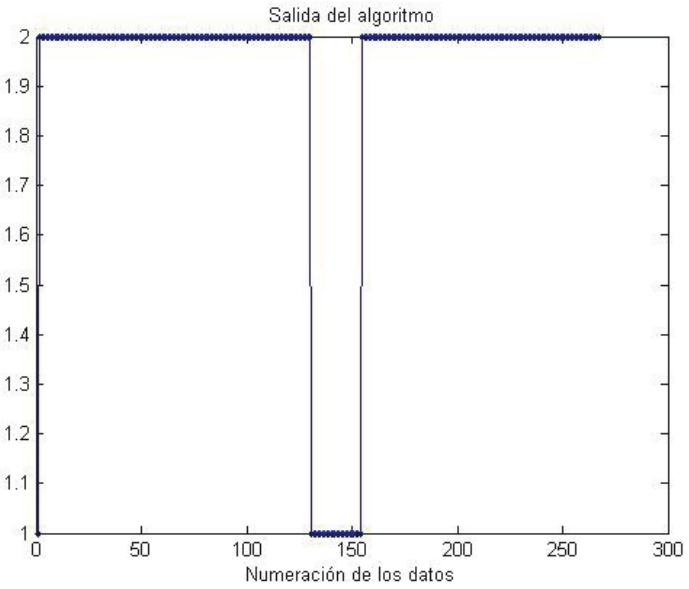


Figura 38A

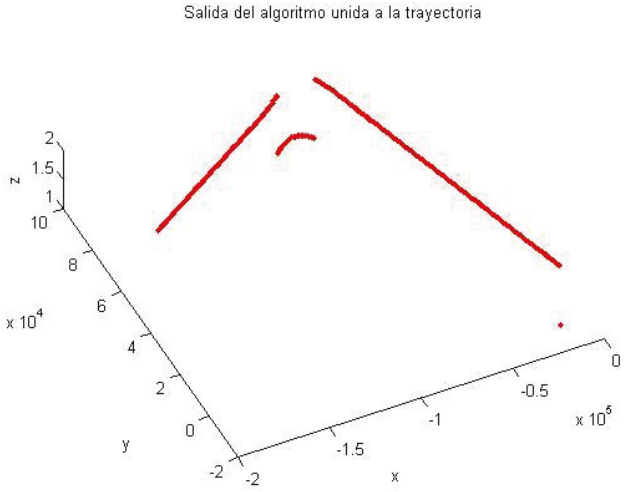


Figura 38B

Tenemos el mismo problema que con la trayectoria 1. En la primera gráfica, que muestra la salida, parece que todo está correcto, pero a la hora de clasificar la curva y colocarla en su lugar retrasa la entrada.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0568181818181818	0,0454545454545455	0
Recto (2)	0,0378787878787879	0,859848484848485	0
G. Dcha. (3)	0	0	0

Tabla 63: Matriz de confusión de la salida con 1/2 de resolución y filtro de Kalman para trayectoria 3

11.4. Datos con 1 de resolución.

Trayectoria 1

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00357142857142857	0,739285714285714	0,0928571428571429
G. Dcha. (3)	0	0,0857142857142857	0,0732142857142857

Tabla 64: Matriz de confusión de la salida con 1 de resolución y filtro de Kalman para trayectoria 1

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 65: Matriz de confusión de la salida con 1 de resolución y filtro de Kalman para trayectoria 2

Trayectoria 3

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0340909090909091	0,0681818181818182	0
Recto (2)	0,0492424242424242	0,848484848484849	0
G. Dcha. (3)	0	0	0

Tabla 66: Matriz de confusión de la salida con 1 de resolución y filtro de Kalman para trayectoria 3

Seguimos viendo resultados muy similares a los anteriores, aunque ligeramente peores, excepto para la trayectoria 2, que la clasifica correctamente, el resto tienen ligeros porcentajes de error debidos a lo que ya hemos comentado, el retraso del filtro a reconocer los cambios de dirección de la trayectoria.

11.5. Datos con 2 de resolución.

Trayectoria 1

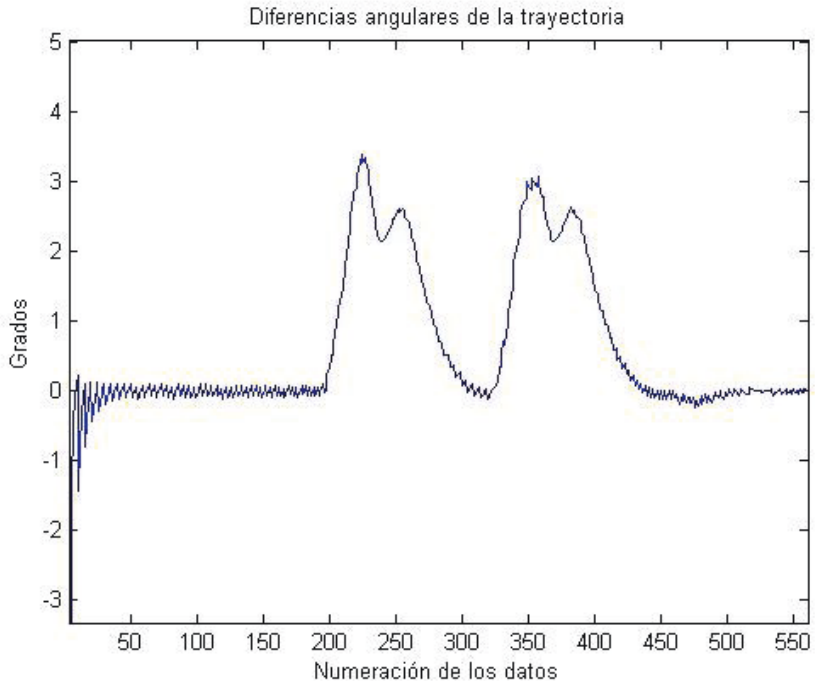


Figura 39A

Como podemos ver, en los datos de entrada las curvas se suavizan demasiado, tanto que al discretizar las diferencias angulares perdemos la pequeña recta existente entre las dos curvas.

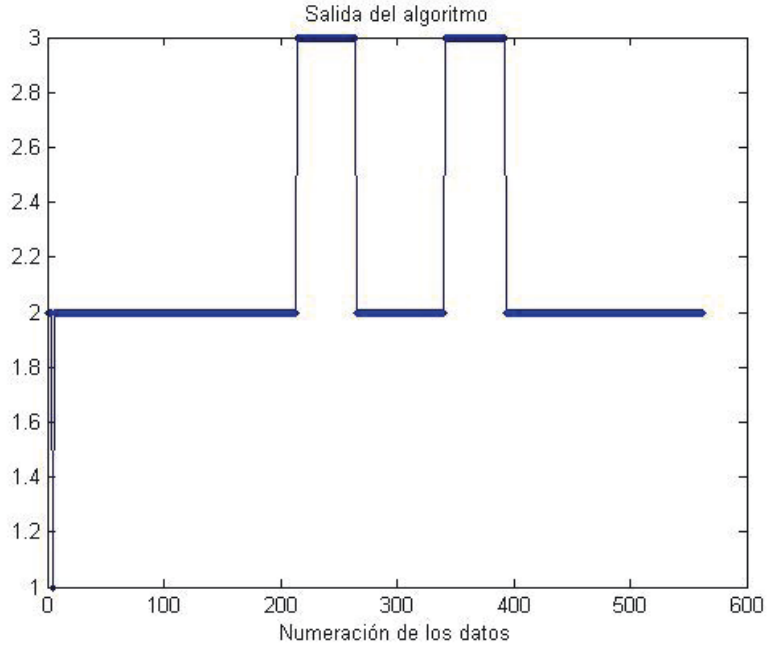


Figura 39B

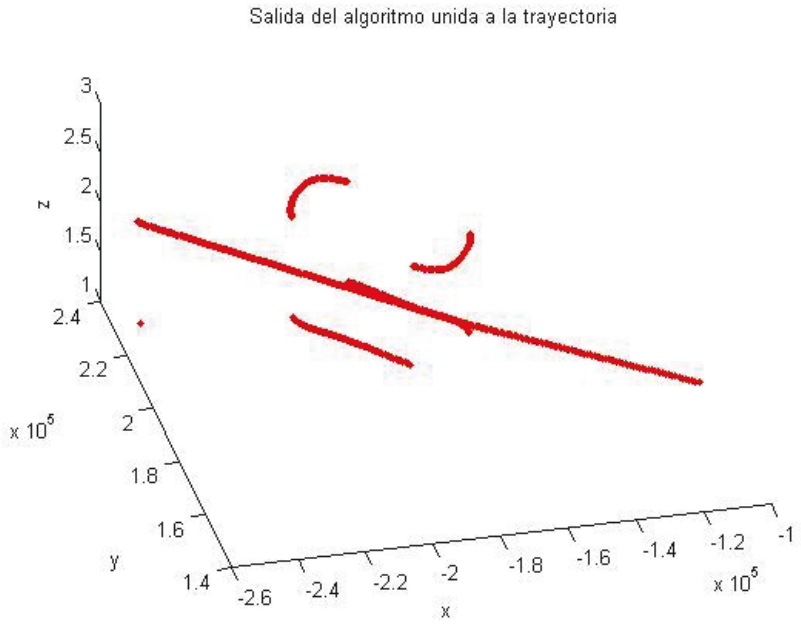


Figura 39C

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00178571428571429	0,733928571428571	0,100000000000000
G. Dcha. (3)	0	0,0732142857142857	0,0857142857142857

Tabla 67: Matriz de confusión de la salida con 2 de resolución y filtro de Kalman para trayectoria 1

Con el gráfico y la matriz podemos apreciar donde comete el algoritmo los errores, pero no se pueden solucionar, ya que los datos obtenidos por el filtro de Kalman no son perfectos.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 68: Matriz de confusión de la salida con 2 de resolución y filtro de Kalman para trayectoria 2

Como ya sabíamos, la salida del algoritmo para la segunda trayectoria sigue siendo perfecta.

Trayectoria 3

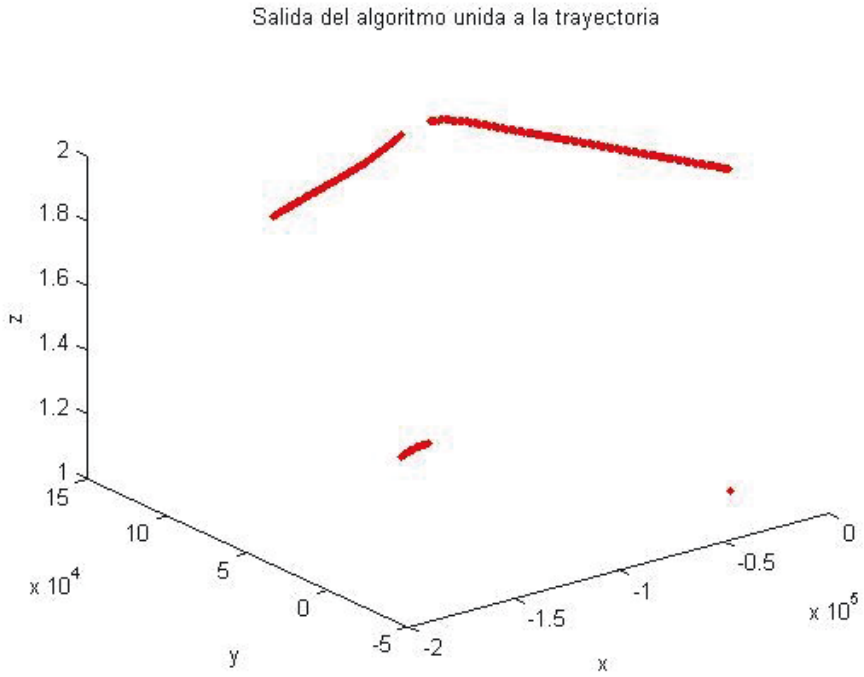


Figura 40

Vemos que sólo un 1% de los datos se clasifican correctamente como giro cuando deberían ser el 7.5% de éstos, y a su vez clasifica un 9% como giro cuando debería ser recto. Aun así clasifica gran parte de las rectas correctamente, y ya que la mayoría de la trayectoria es recta consigue un buen porcentaje.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0113636363636364	0,0909090909090909	0
Recto (2)	0,0643939393939394	0,833333333333333	0
G. Dcha. (3)	0	0	0

Tabla 69: Matriz de confusión de la salida con 2 de resolución y filtro de Kalman para trayectoria 3

11.6. Comparación de aciertos con distintos niveles de resolución:

Trayectorias / Porcentaje de resolución	1	2	3	Media
1/8	89.6%	99%	95.8%	**94.81%**
1/4	87%	100%	94%	**93.63%**
1/2	84.8%	100%	91.6%	**92.16%**
1	81.2%	100%	88.2%	**89.83%**
2	82%	100%	84.4%	**88.81%**

Tabla 70: Porcentaje de aciertos del Viterbi con distintos grados de resolución sobre trayectorias con filtro de Kalman

A pesar de los problemas con el filtro de Kalman, los datos son bastante buenos, incluso mejores que sin éste, como era de esperar ya que el filtro de Kalman está pensado para suavizar los efectos del ruido y de la resolución.

12. Modelo con datos simulados con ruido, sin resolución y filtro de Kalman

Como hemos podido observar, anteriormente el ruido afectaba de una manera negativa a los datos y a la salida del algoritmo, por lo que esperamos que con el filtro de Kalman se pueda mitigar, y así obtener unos mejores resultados.

12.1. Datos con 1/8 de ruido.

Trayectoria 1

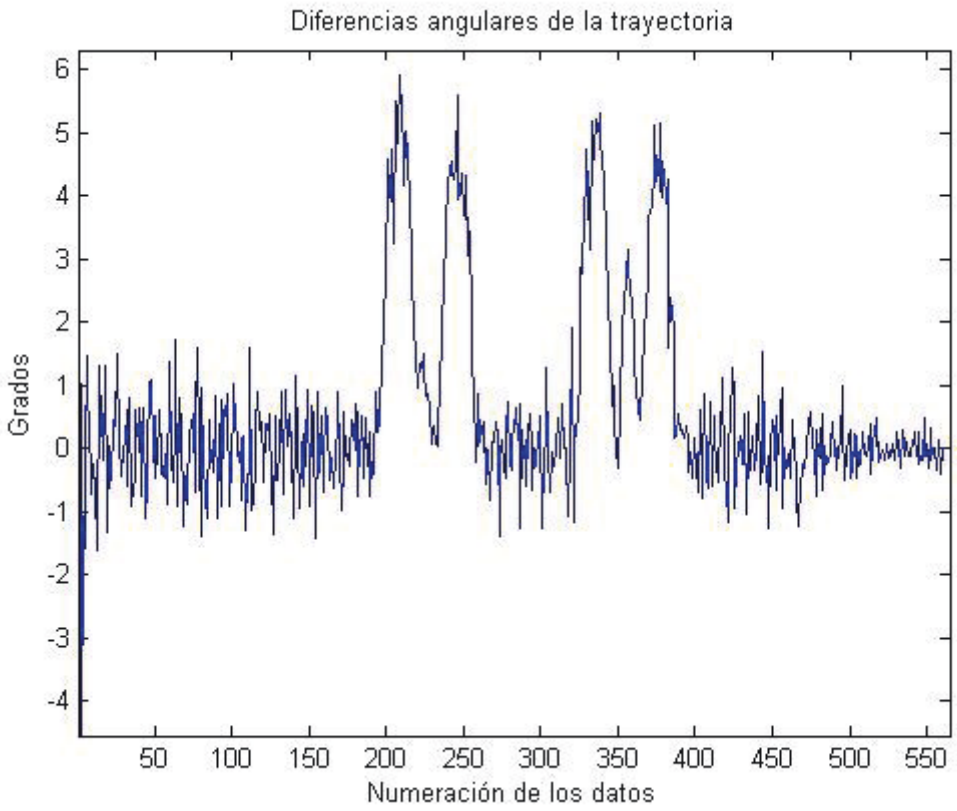


Figura 41A

Vemos como los datos de entrada tienen mitigado el ruido por el filtro de Kalman, lo que nos permite apreciar los giros mediante las diferencias angulares.

Si lo comparamos con los datos obtenidos que ese porcentaje de ruido pero sin el filtro, veremos la diferencia.

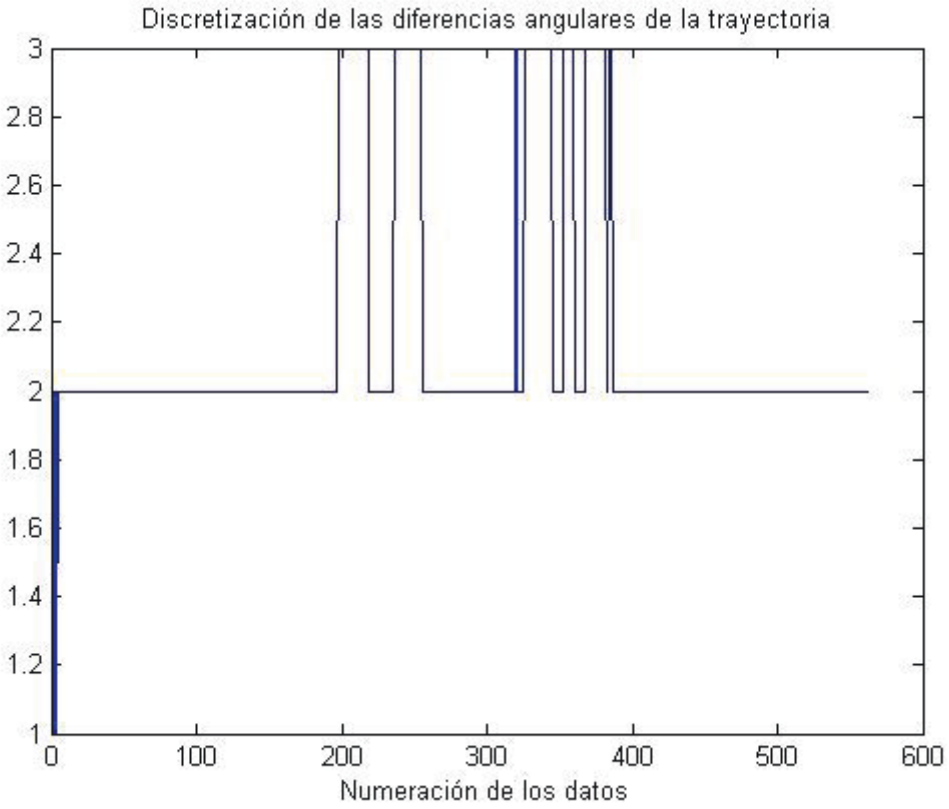


Figura 41B

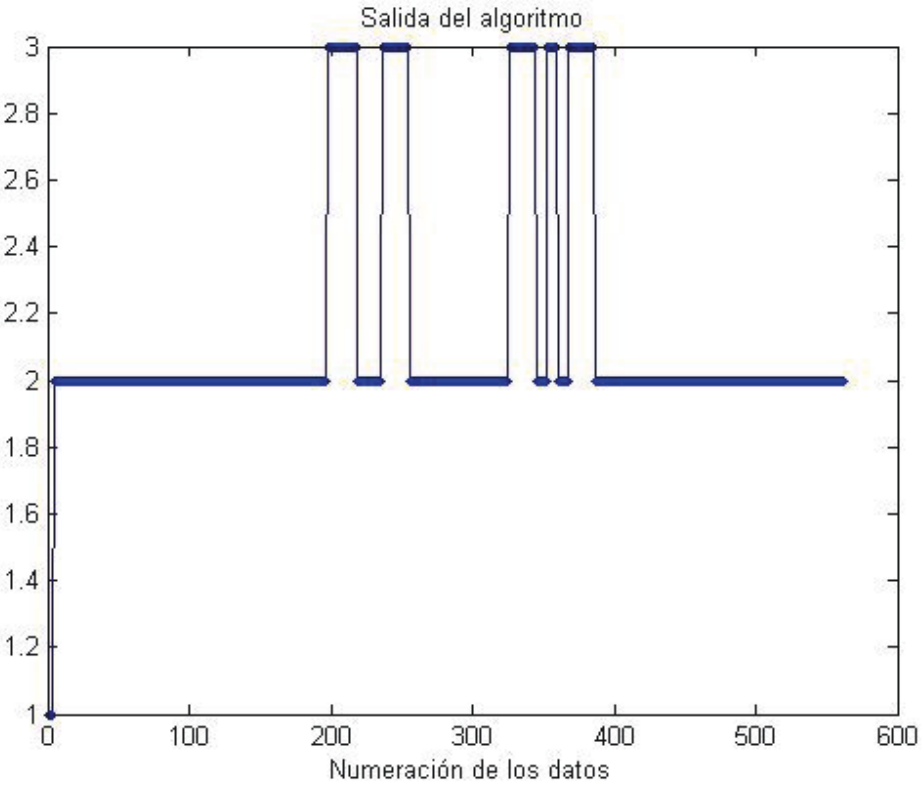


Figura 41C

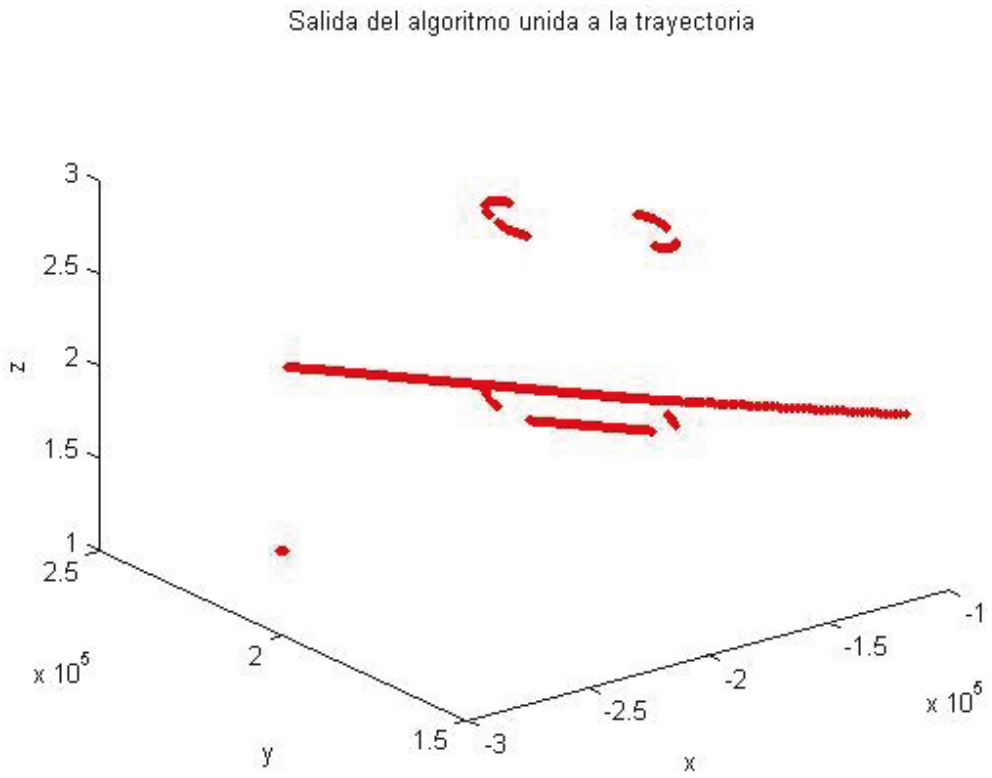


Figura 41D

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00714285714285714	0,785714285714286	0,0428571428571429
G. Dcha. (3)	0	0,0482142857142857	0,110714285714286

Tabla 71: Matriz de confusión de la salida con 1/8 de ruido y filtro de Kalman para trayectoria 1

Como hemos visto, la discretización se ha realizado correctamente y la salida del algoritmo, exceptuando algunos pequeños fallos, es correcta.

Trayectoria 2

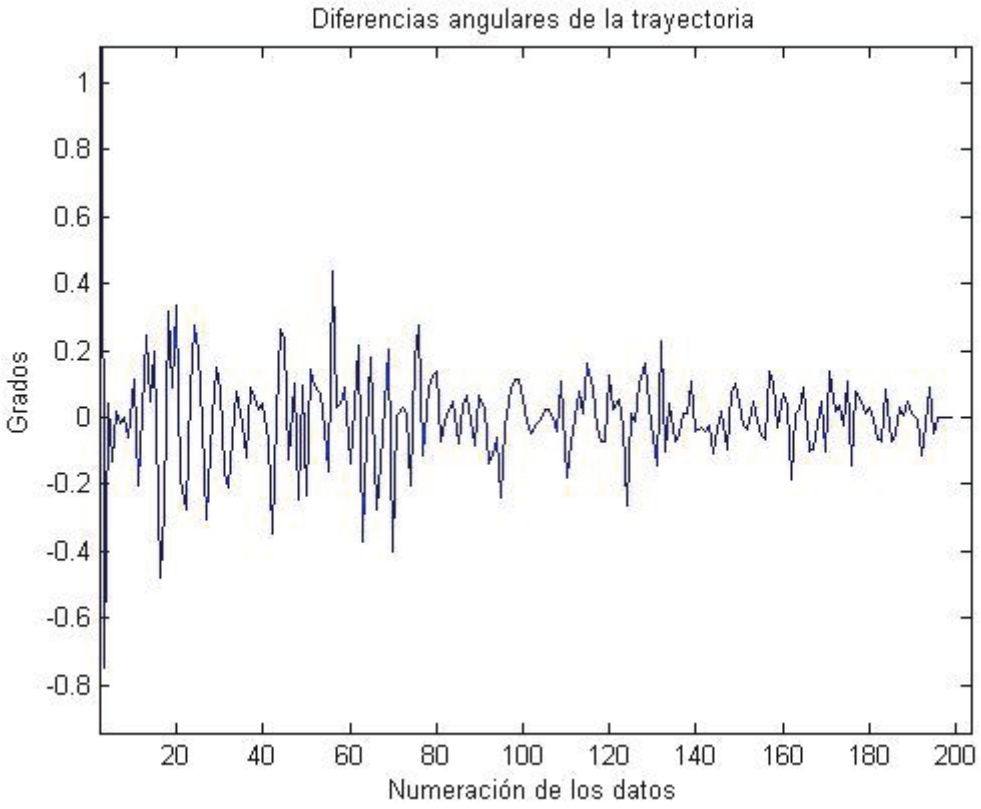


Figura 42

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	0,989795918367347	0,0102040816326531
G. Dcha. (3)	0	0	0

Tabla 72: Matriz de confusión de la salida con 1/8 de ruido y filtro de Kalman para trayectoria 2

El filtro de Kalman, como hemos mencionado anteriormente, donde mejor disipa el ruido es en las rectas, por lo que las diferencias angulares de la trayectoria 2 son mínimas, y esto permite al algoritmo hacer una excelente clasificación.

Trayectoria 3

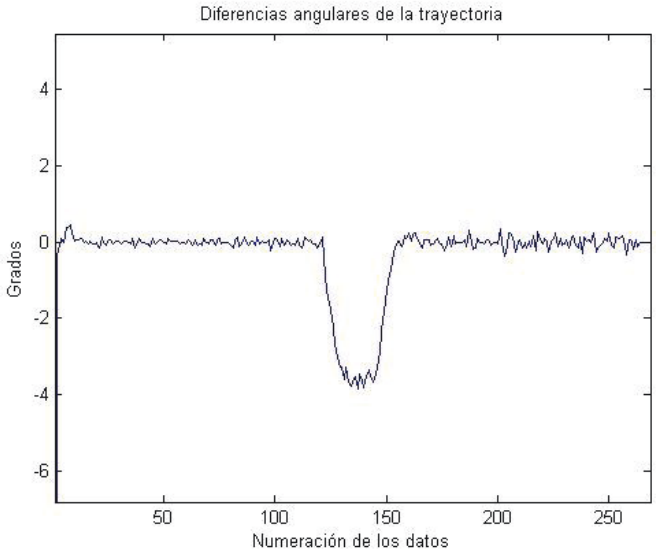


Figura 43A

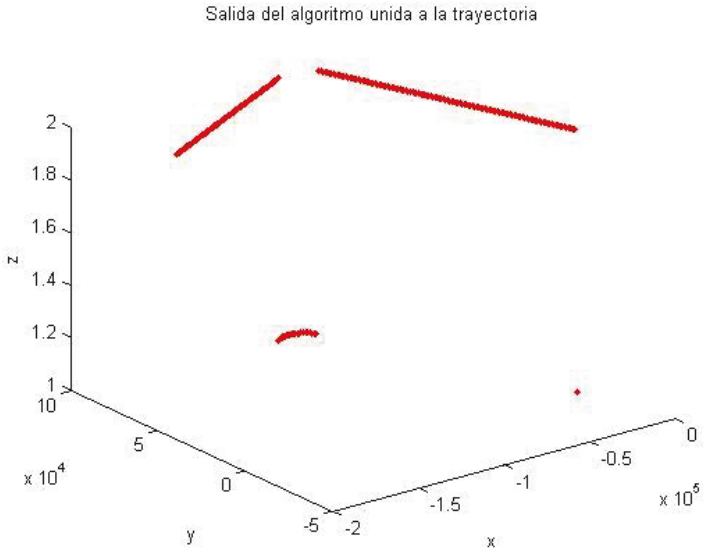


Figura 43B

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0757575757575758	0,0265151515151515	0
Recto (2)	0,0189393939393939	0,878787878787879	0
G. Dcha. (3)	0	0	0

Tabla 73: Matriz de confusión de la salida con 1/8 de ruido y filtro de Kalman para trayectoria 3

En la trayectoria 3, ya podemos apreciar entorno al giro, cómo el filtro lo retrasa, y por eso aparecen esos pequeños errores que en total no suman ni un 5% de la trayectoria.

12.2. Datos con 1/4 de ruido.

Trayectoria 1

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00535714285714286	0,771428571428572	0,0589285714285714
G. Dcha. (3)	0	0,0535714285714286	0,105357142857143

Tabla 74: Matriz de confusión de la salida con 1/4 de ruido y filtro de Kalman para trayectoria 1

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	0,989795918367347	0,0102040816326531
G. Dcha. (3)	0	0	0

Tabla 75: Matriz de confusión de la salida con 1/4 de ruido y filtro de Kalman para trayectoria 2

Trayectoria 3

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0719696969696970	0,0303030303030303	0
Recto (2)	0,0265151515151515	0,871212121212121	0
G. Dcha. (3)	0	0	0

Tabla 76 Matriz de confusión de la salida con 1/4 de ruido y filtro de Kalman para trayectoria 2

Los resultados de las tres trayectorias son muy similares a los anteriores, en comparación a no usar el filtro de Kalman los resultados son increíbles. Se ha producido una mejora sustancial a la hora de realizar las clasificaciones, y no sólo para una trayectoria en concreto, sino para las tres.

12.3. Datos con 1/2 de ruido.

A medida que aumentamos el ruido, como ya hemos visto con los datos con resolución, el filtro de Kalman funciona peor, y añade un retardo cada vez mayor.

No obstante, seguimos apreciando la mejora respecto a los datos con ruido y sin filtro de Kalman, ya que a estos niveles de ruido, el algoritmo no era capaz de realizar una clasificación mejor que coger la acción más numerosa.

Trayectoria 1

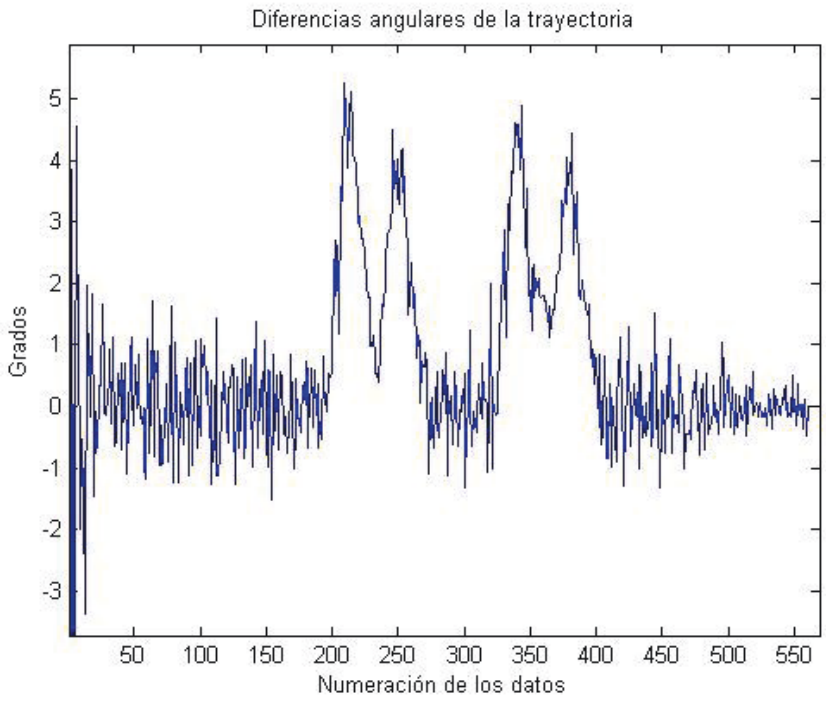


Figura 44

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00892857142857143	0,769642857142857	0,0571428571428571
G. Dcha. (3)	0	0,0589285714285714	0,1

Tabla 77: Matriz de confusión de la salida con 1/2 de ruido y filtro de Kalman para trayectoria 1

Como vemos, los datos antes de realizar la discretización son muy claros. No obstante, tenemos entorno a un 10% de error, ya que el filtro de Kalman retrasa el reconocimiento de los giros al eliminar el ruido.

Trayectoria 2

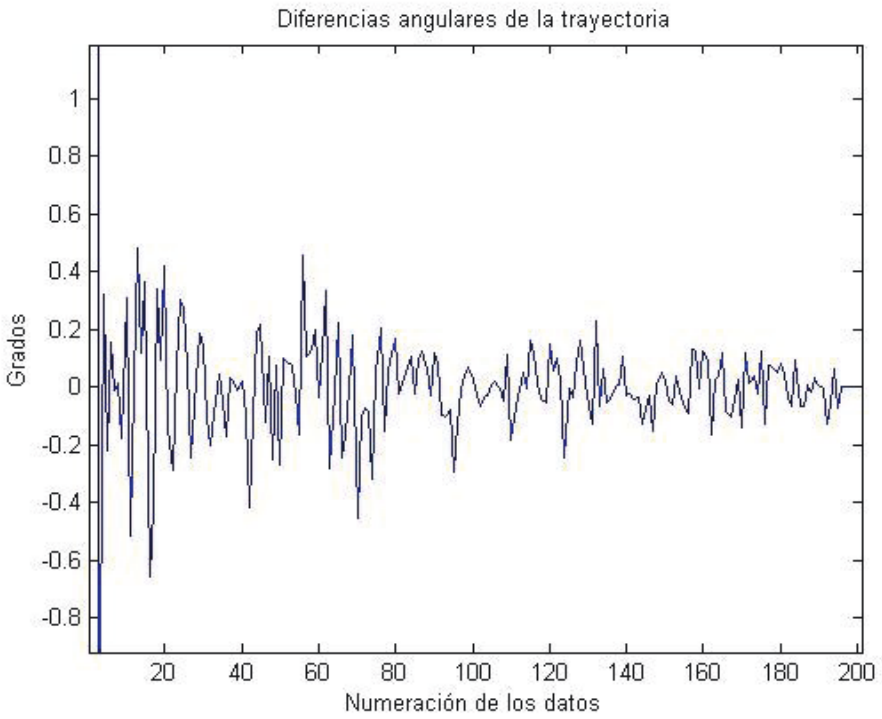


Figura 45

Como vemos, los datos de entrada no tienen prácticamente nada de ruido. La diferencia angular en casi la totalidad de la trayectoria es menor a un grado.

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	0,989795918367347	0,0102040816326531
G. Dcha. (3)	0	0	0

Tabla 78: Matriz de confusión de la salida con 1/2 de ruido y filtro de Kalman para trayectoria 2

Al eliminar la mayor parte del ruido y ser una recta el filtro de Kalman, y el HMM junto con el algoritmo, funcionan a la perfección.

Trayectoria 3

A nuestra derecha, tenemos el gráfico de la trayectoria ideal, y a nuestra izquierda tenemos el de la trayectoria con 1/2 de ruido con filtro de Kalman.

Parecen exactamente iguales, pero si nos fijamos en la numeración de los datos, en el primer gráfico la curva se inicia sobre el 115 y termina sobre el 145, mientras que en el segundo vemos que va desde el 135 al 155, aparte de reducir la curva se ha desplazado ligeramente a la derecha (retardo). Esto es el efecto que produce el filtro. No obstante, vemos que se trata de un resultado muy real.

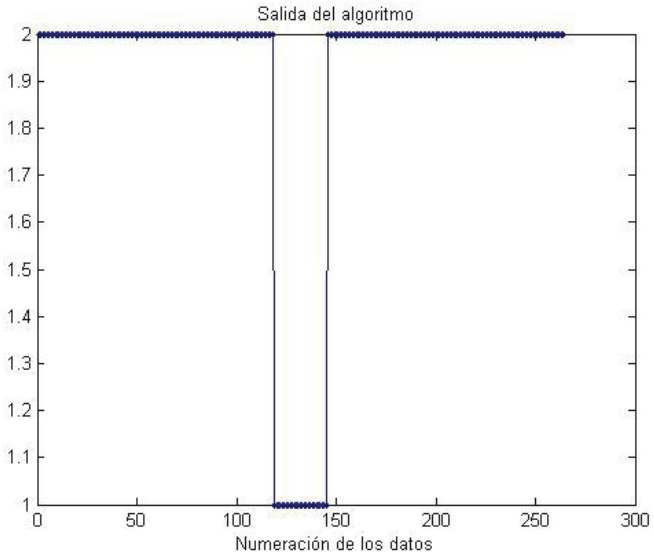


Figura 46A

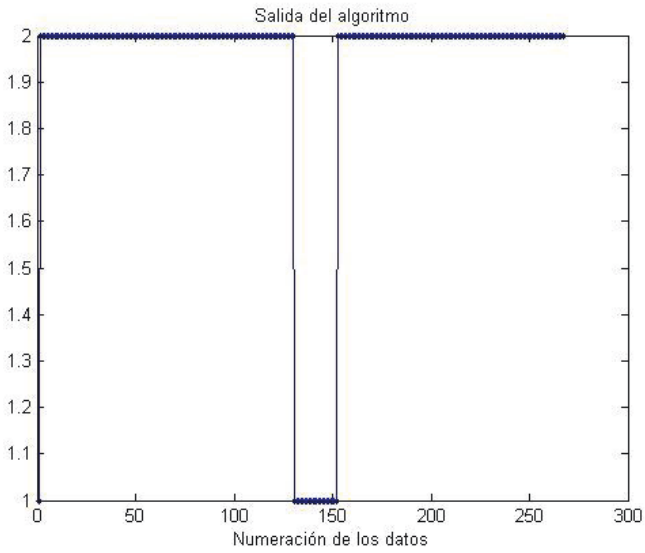


Figura 46B

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0568181818181818	0,0454545454545455	0
Recto (2)	0,0303030303030303	0,867424242424242	0
G. Dcha. (3)	0	0	0

Tabla 79: Matriz de confusión de la salida con 1/2 de ruido y filtro de Kalman para trayectoria 3

12.4. Datos con 1 de ruido.

Trayectoria 1

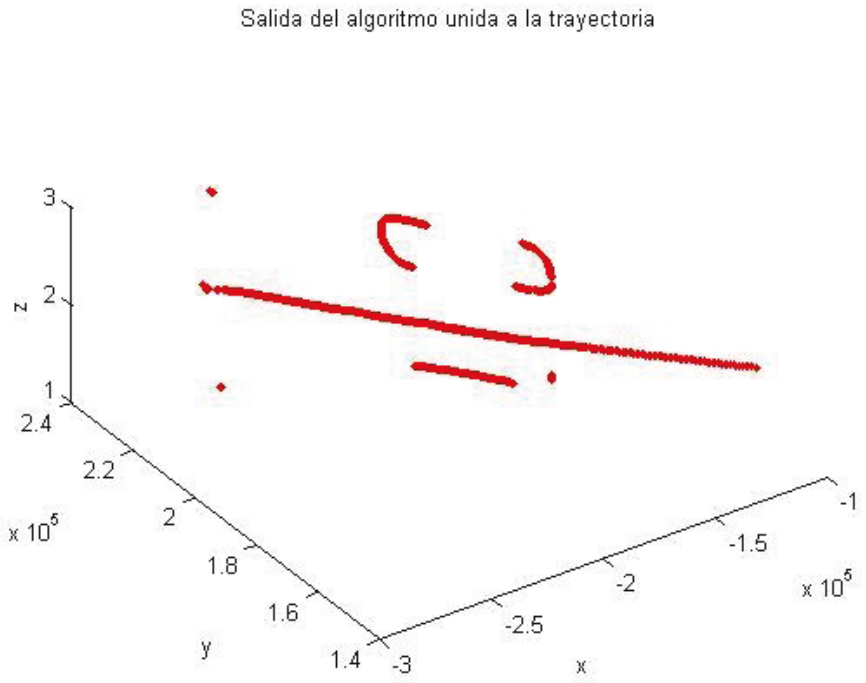


Figura 47

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00357142857142857	0,710714285714286	0,121428571428571
G. Dcha. (3)	0	0,0375000000000000	0,121428571428571

Tabla 80: Matriz de confusión de la salida con 1 de ruido y filtro de Kalman para trayectoria 1

Al suavizar el efecto del ruido, también se suavizan las curvas, lo que conlleva que pierda las pequeñas curvas cuanto más ruido tengamos. Ese es el motivo por el que en el gráfico ya vemos dos grandes curvas, en vez de ver cuatro de la mitad de tamaño, debido a que las unifica.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 81: Matriz de confusión de la salida con 1 de ruido y filtro de Kalman para trayectoria 2

Como ya hemos dicho, donde mejor funciona es en las rectas.

Trayectoria 3

Vemos como se retrasa aun más la curva, lo que provoca un ligero aumento de los fallos.

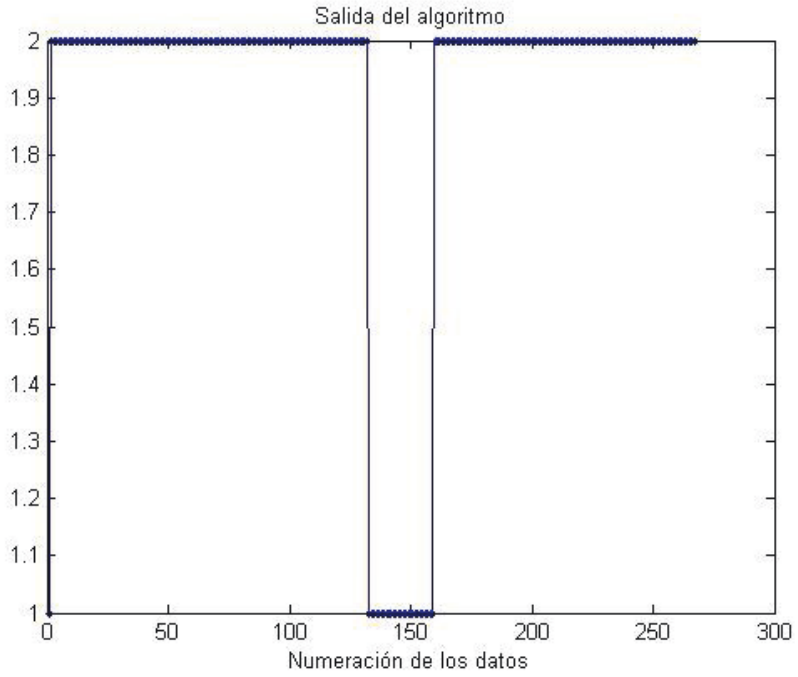


Figura 48

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,0492424242424242	0,0530303030303030	0
Recto (2)	0,0568181818181818	0,840909090909091	0
G. Dcha. (3)	0	0	0

Tabla 82: Matriz de confusión de la salida con 1 de ruido y filtro de Kalman para trayectoria 3

12.5. Datos con 2 de ruido.

Trayectoria 1

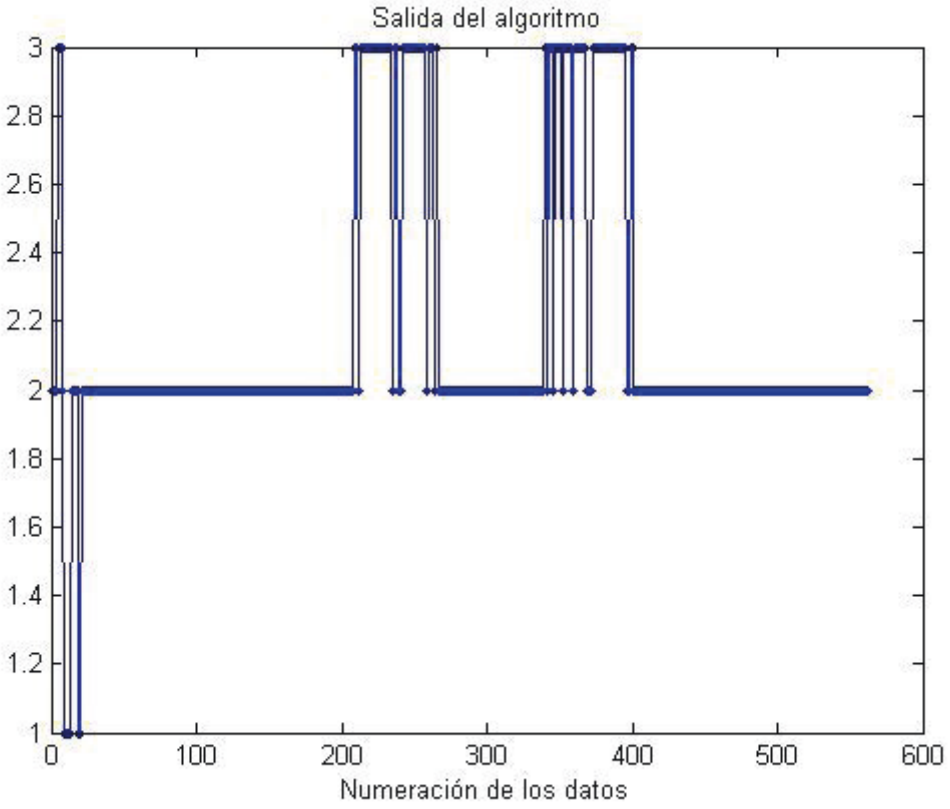


Figura 49

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0,00535714285714286	0
Recto (2)	0,00892857142857143	0,705357142857143	0,121428571428571
G. Dcha. (3)	0	0,0750000000000000	0,0839285714285714

Tabla 83: Matriz de confusión de la salida con 2 de ruido y filtro de Kalman para trayectoria 1

Como vemos, la salida no es perfecta, pero siendo éste el peor caso de las trayectorias simuladas, obtiene un resultado muy bueno gracias al filtro de Kalman.

Trayectoria 2

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0	0	0
Recto (2)	0	1	0
G. Dcha. (3)	0	0	0

Tabla 84: Matriz de confusión de la salida con 2 de ruido y filtro de Kalman para trayectoria 2

Trayectoria 3

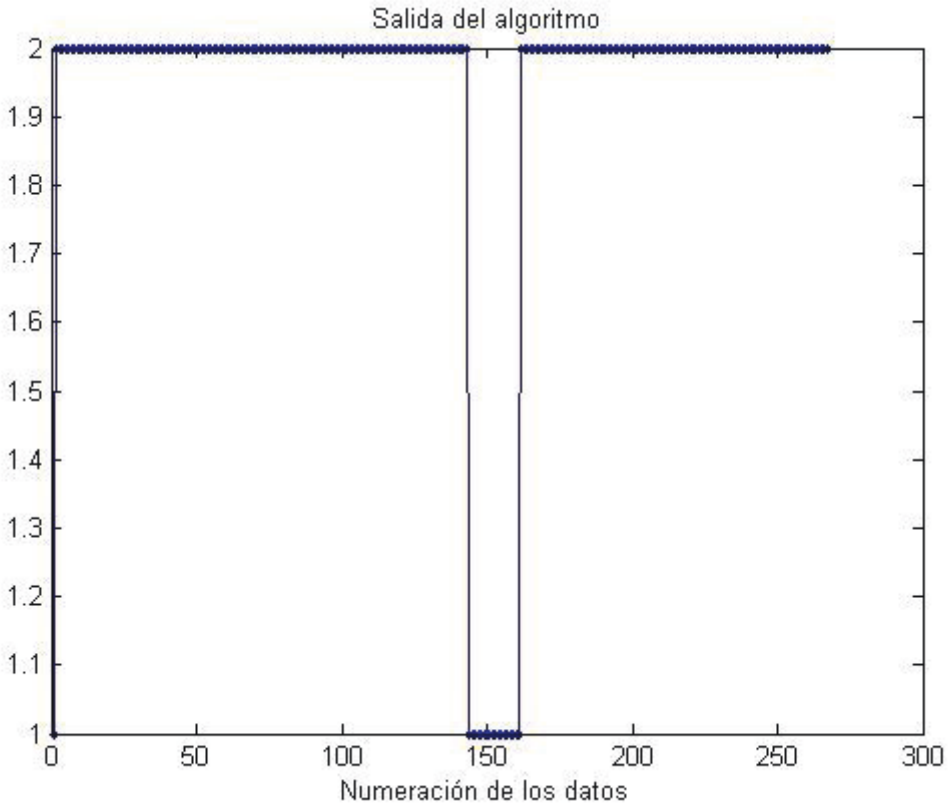


Figura 50

Predicción / Real	G. Izq. (1)	Recto (2)	G. Dcha. (3)
G. Izq. (1)	0,00757575757575758	0,0946969696969697	0
Recto (2)	0,0643939393939394	0,833333333333333	0
G. Dcha. (3)	0	0	0

Tabla 85: Matriz de confusión de la salida con 2 de ruido y filtro de Kalman para trayectoria 3

12.6. Comparación de aciertos con distintos niveles de ruido:

Trayectorias / Porcentaje de ruido	1	2	3	Media
1/8	89.6%	99%	95.4%	**94.67%**
1/4	87.7%	99%	94.3%	**93.67%**
1/2	87%	99%	92.4%	**92.8%**
1	83.2%	100%	89%	**90.73%**
2	79%	100%	84%	**87.67%**

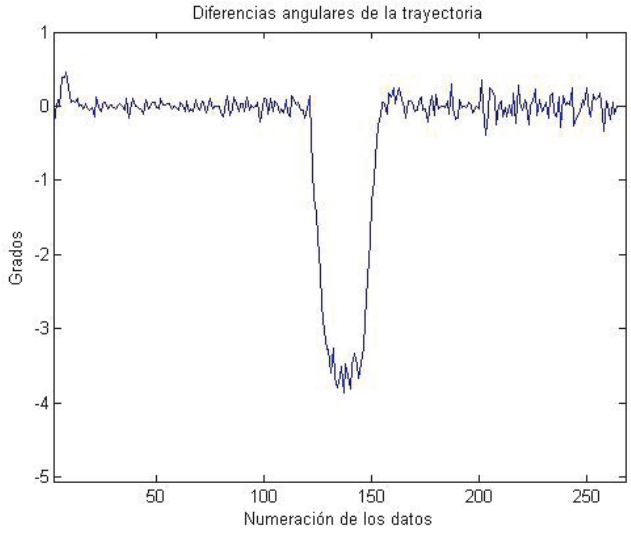
Tabla 86: Porcentaje de aciertos del Viterbi con distintos grados de ruido sobre trayectorias con filtro de Kalman

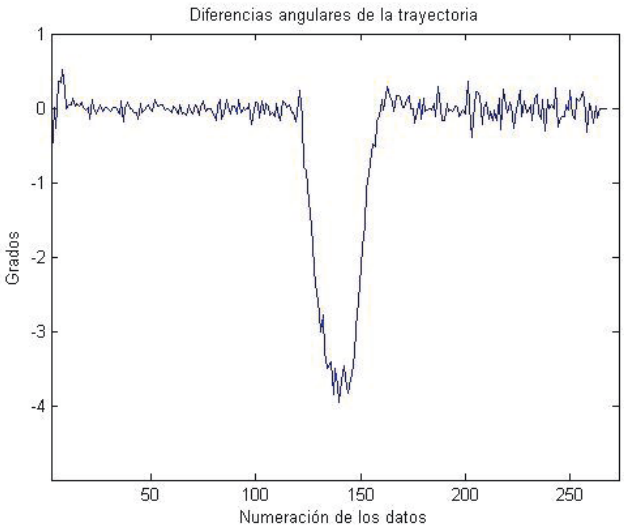
Comparación basada en el filtro de Kalman para la trayectoria 3

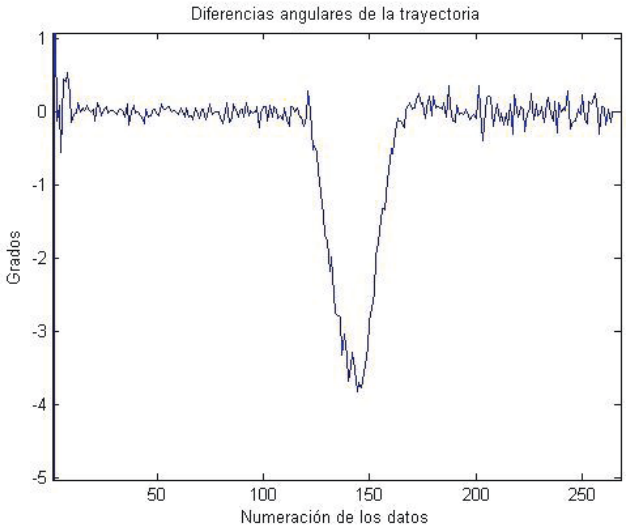
Hemos ido apreciando cómo el filtro de Kalman permitía disminuir ligeramente el ruido, y cómo afectaba a los datos de entrada, y más tarde a la salida del algoritmo.

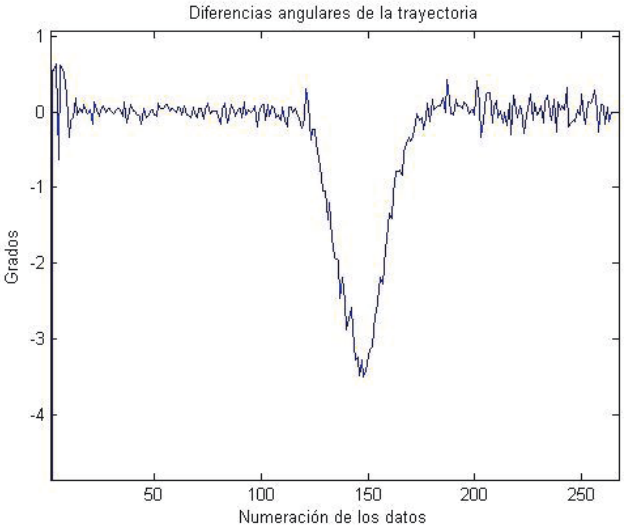
Vamos a observar la evolución de la trayectoria 3 en concreto, que nos permite ver claramente el problema del filtro.

Tenemos los datos de entrada con 1/8, 1/4, 1/2, 1 y 2 de ruido en las figuras *51A, 51B, 51C, 51D, 51E*; respectivamente.









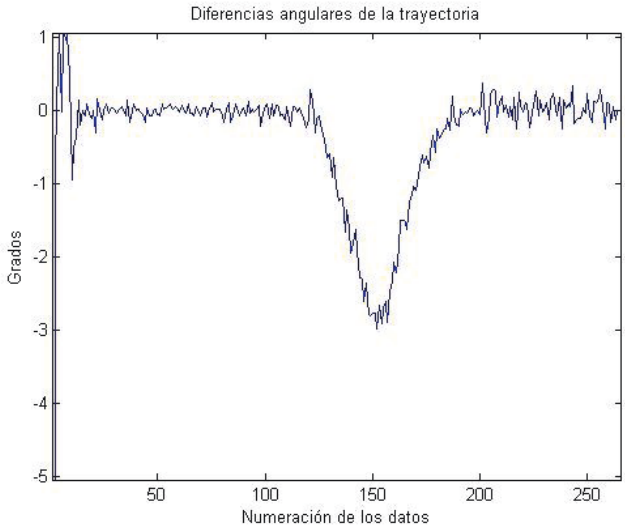
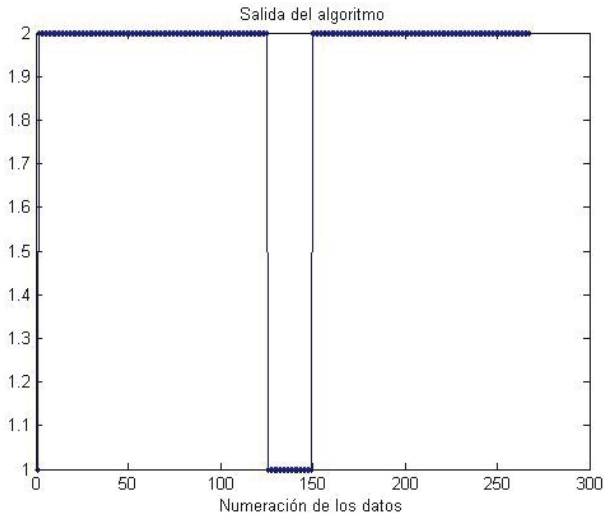
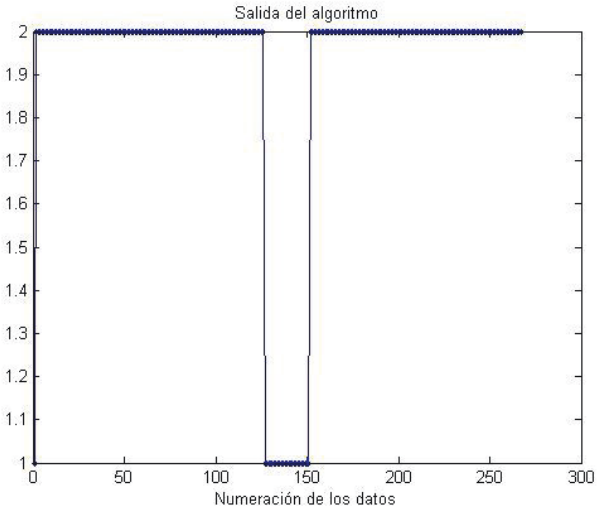


Figura 51A,B,C,D,E

Es totalmente visible cómo en la primera grafica (*51A*) la diferencia angular es mayor en los datos inferiores a 150, mientras que según aumenta el ruido hasta la última grafica (51*E*), la diferencia angular mayor (curva) se encuentra en los datos superiores a 150, lo que nos indica que el retraso no lo produce el algoritmo ni el modelo, como ya sabíamos, sino que viene producido por el filtro de Kalman. Lo que provoca que la salida del algoritmo para los distintos grados de ruido sea la siguiente;





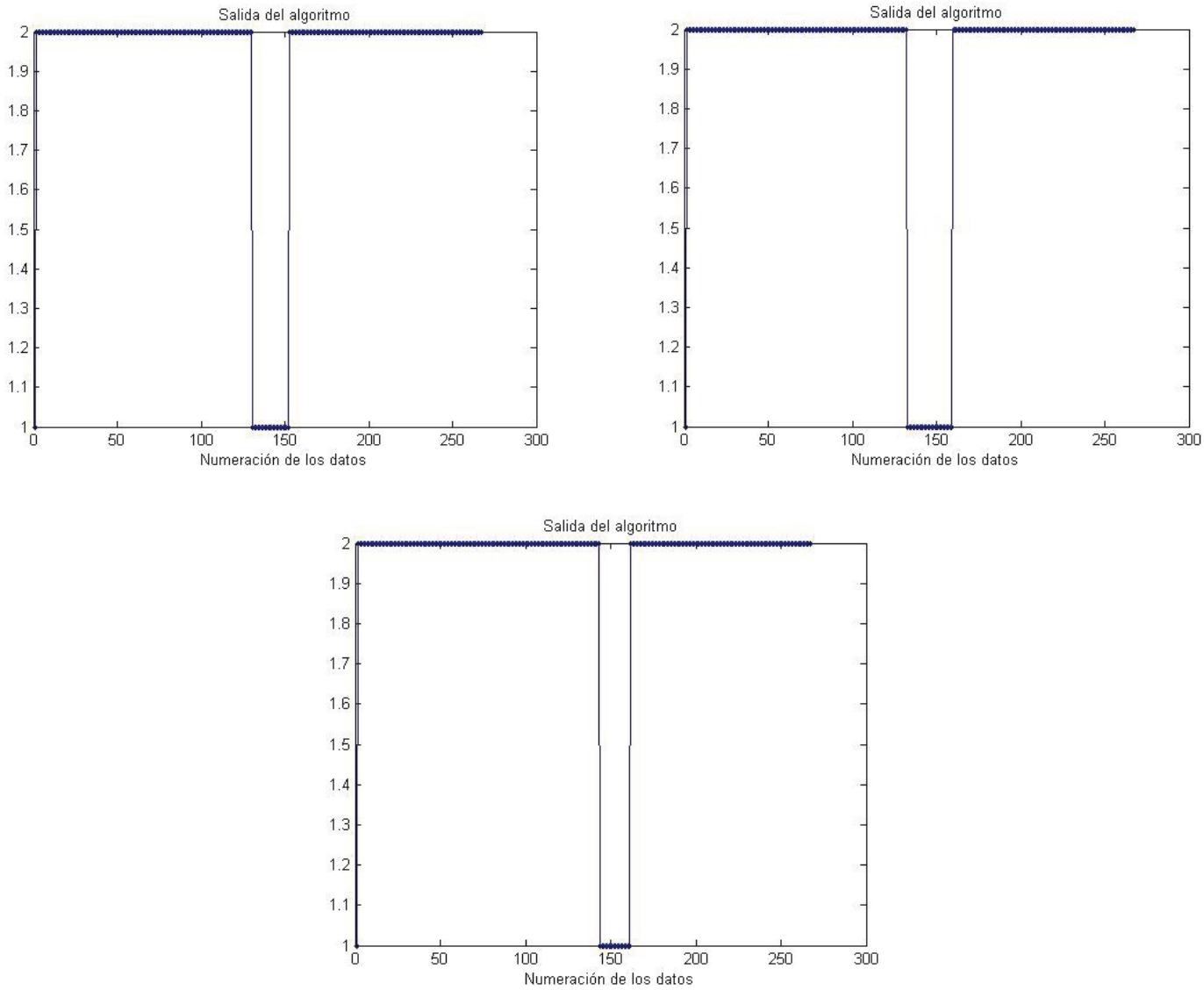


Figura 52A,B,C,D,E

Podemos ver la misma progresión o retraso en la salida del algoritmo, ya que no tiene capacidad ninguna para diferenciar si la curva se está retrasando por el filtro, o simplemente empieza más tarde.

13. Comparación de resultados para las trayectorias simuladas

Haciendo una media entre las tres trayectorias, obtenemos esta tabla para con los distintos parámetros con los que se han realizado las simulaciones.

Tipos Porcentajes	Resolución	Ruido	Resolución y filtro de Kalman	Ruido y filtro de Kalman
1/8	99.26%	32.26%	94.81%	94.67%
1/4	90.2%	38.1%	93.63%	93.67%
1/2	86.63%	87.8%	92.16%	92.8%
1	73.8%	90.5%	89.83%	90.73%
2	73.2%	90.7%	88.81%	87.67%

Tabla 87: Porcentaje de aciertos medio de todas las trayectorias simuladas divididas por sus parámetros

Primero, vemos cómo el aumento de la resolución en las trayectorias simuladas tiene un efecto que perjudica la salida hasta con un 30% de fallos en la clasificación de ésta. Apreciamos un pequeño escalón en la progresión de las medias con resolución 1, pero por lo demás es normal.

Después, con el ruido, el algoritmo no obtiene ningún resultado decente, el elevado grado de aciertos no tienen ningún valor en la clasificación, ya que sólo realiza la acción más numerosa, y dado que la más numerosa está presente aproximadamente en un 90% de las trayectorias, consigue ese porcentaje. No obstante, si se modificaran las trayectorias e incluyésemos más giros el porcentaje disminuiría drásticamente.

Con resolución y un filtro de Kalman aplicado a los datos de entrada, vemos cómo al principio el resultado de la clasificación es peor, ya que el filtro de Kalman está pensado para suavizar los saltos en los datos producidos por ruido o resolución, y cuando éstos son muy pequeños, el algoritmo funciona mejor con los datos originales. Sin embargo, vemos que el aumento de los aciertos no se hace esperar cuando aumentamos la resolución.

Para el ruido y con el filtro de Kalman, vemos como se produce un cambio sorprendente. Pasa de prácticamente no poder realizar la clasificación, a realizar una clasificación francamente buena, con un 87% de aciertos en el peor caso, lo que está claro que el algoritmo trabaja mejor con los datos resultantes al usar el filtro de Kalman que simplemente un ruido demasiado elevado.

14. Modelo con datos reales

Para realizar la clasificación con datos reales, hemos elegido una serie de trayectorias que consideramos que pueden dar más juego a la hora de analizar la eficacia del algoritmo con cualquier situación real. Primero usaremos los datos reales en crudo y a continuación los datos reales después de aplicar el filtro de Kalman.

En este apartado realizaremos un análisis a mayor profundidad de los datos y los resultados de cada una de las trayectorias. Al no ser datos simulados no tenemos las trayectorias ideales, por lo que tendremos que prescindir de la matriz de confusión.

Usaremos las siguientes trayectorias:

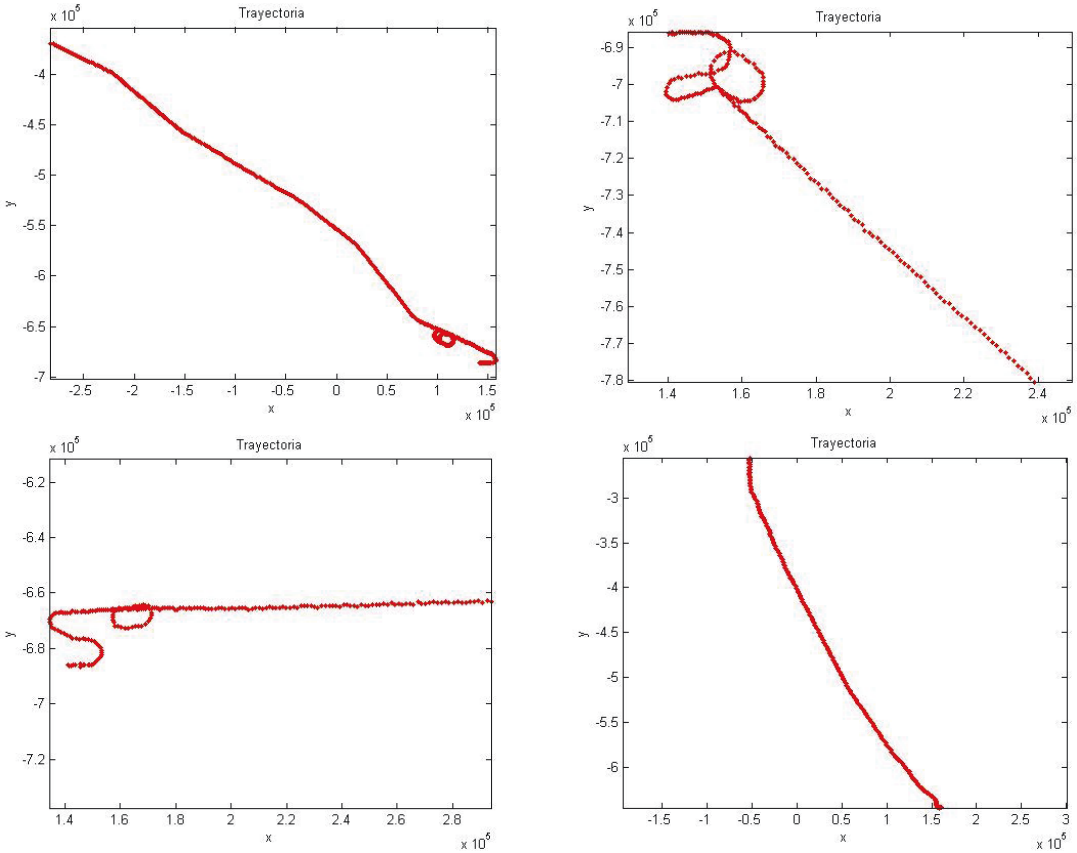


Figura 8A, B, C, D: Trayectorias reales

14.1. Trayectorias reales sin aplicar el filtro de Kalman

El modelo entrenado para clasificar las trayectorias reales es el siguiente:

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.55	0.35	0.1
Recto (2)	0.15	0.7	0.15
Giro Dcha. (3)	0.1	0.35	0.55

Tabla 88: Matriz de transición del modelo para datos reales

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.55	0.4	0.05
Diferencia angular pequeña (2)	0.2	0.6	0.2
Diferencia angular grande positiva (3)	0.05	0.4	0.55

Tabla 89: Matriz de observación del modelo para datos reales

Los topes usados en la discretización de los datos son 1.5 y -1.5

Trayectoria 1

Como hemos podido ver, la trayectoria 1 tiene unos giros muy ligeros al principio y más tarde realiza los giros más pronunciados.

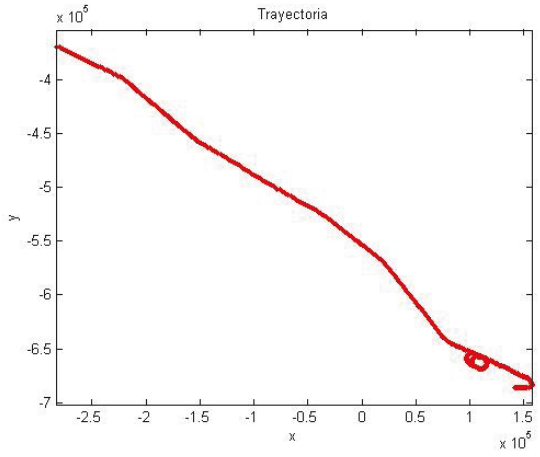


Figura 8A

Vamos a mostrar los tamaños de los ángulos que forman la unión de los puntos de las trayectorias.

Vemos como las diferencias de los puntos son mínimas, pasado el punto 250 empiezan a aumentar progresivamente hasta que vuelven a descender y aumentan al final de nuevo. Esto se debe a los giros, se ve claramente como desde el 250 hasta el 325 aproximadamente el avión está girando.

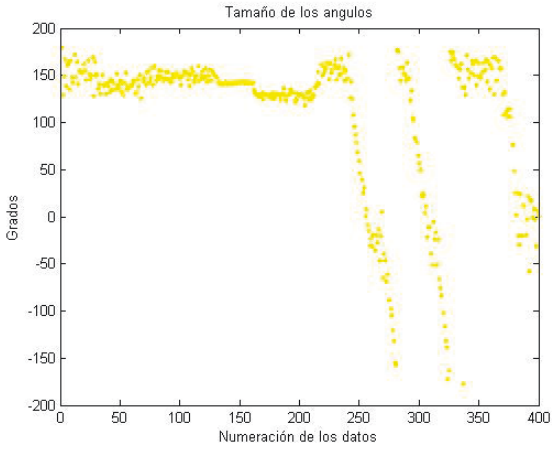


Figura 53A

A partir del tamaño de los ángulos, sacamos las diferencias angulares y obtenemos las siguientes gráficas.

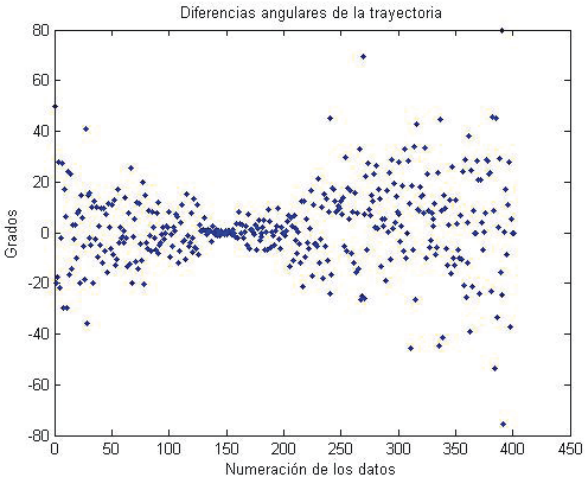


Figura 53B

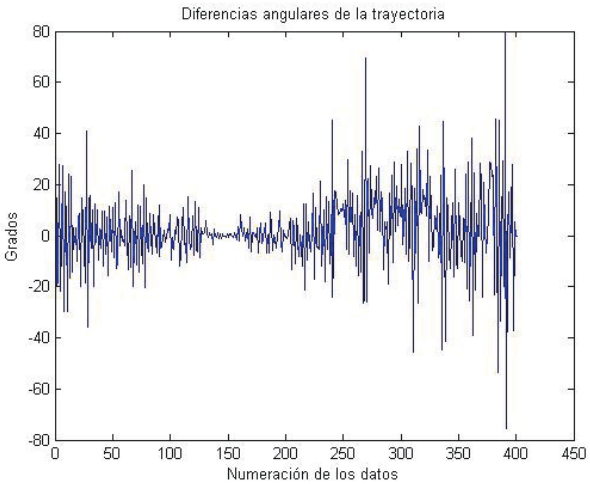


Figura 53C

Las dos son la misma gráfica representada en dos formatos distintos. Por supuesto, se observa que el ruido en el centro de la trayectoria es menor que en los extremos, posiblemente debido a la distancia al sensor. Y si nos fijamos en la gráfica de la derecha, podemos observar las pequeñas curvas, en la parte inferior. Una vez discretizamos las diferencias angulares, tenemos el siguiente resultado. Estos datos serán los datos de entrada para el algoritmo junto con el modelo.

Como vemos, si quisiéramos aplicar la discretización como solución al problema, no sería del todo efectiva, ya que los datos de las diferencias angulares no permiten conocer las acciones del avión exactamente.

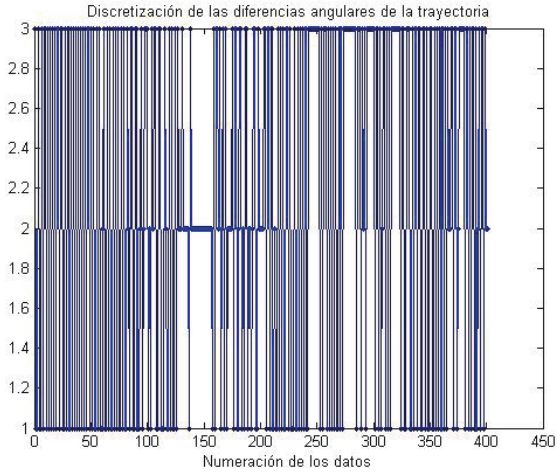


Figura 53D

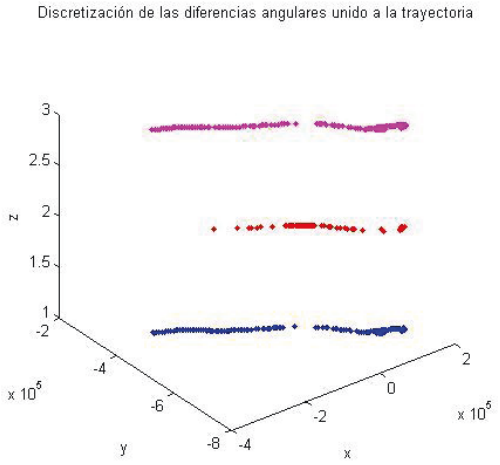


Figura 53E

Obtenemos la salida del algoritmo, que es la siguiente:

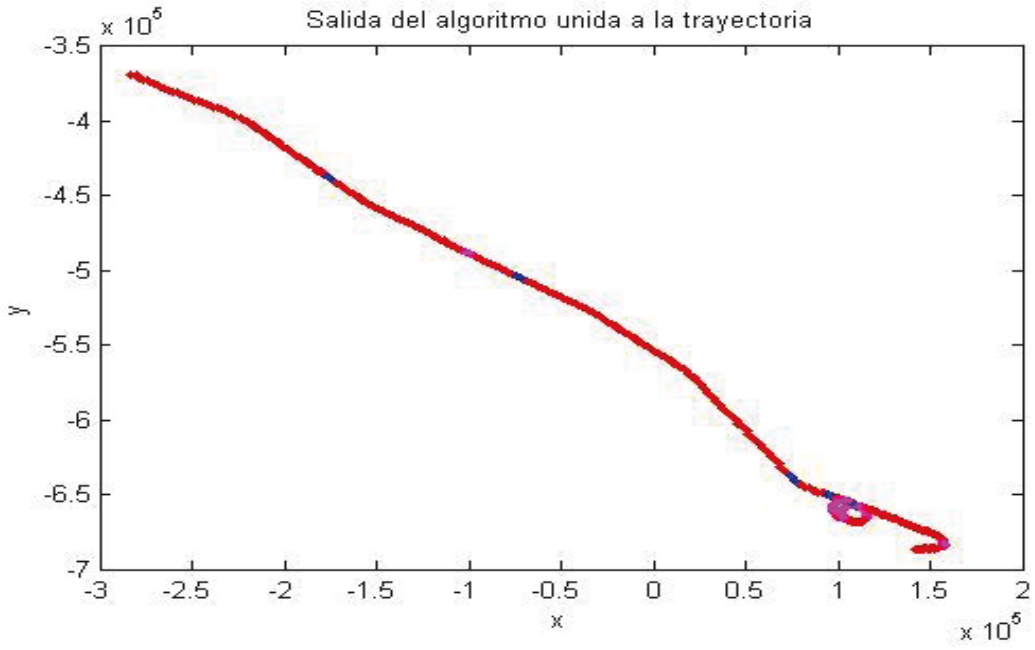


Figura 53F

Si observamos las gráficas, se consigue distinguir pequeñas curvas a lo largo de la trayectoria pero sin mucho éxito. Pero vamos a centrarnos en el final de la trayectoria, donde se acumulan las curvas:

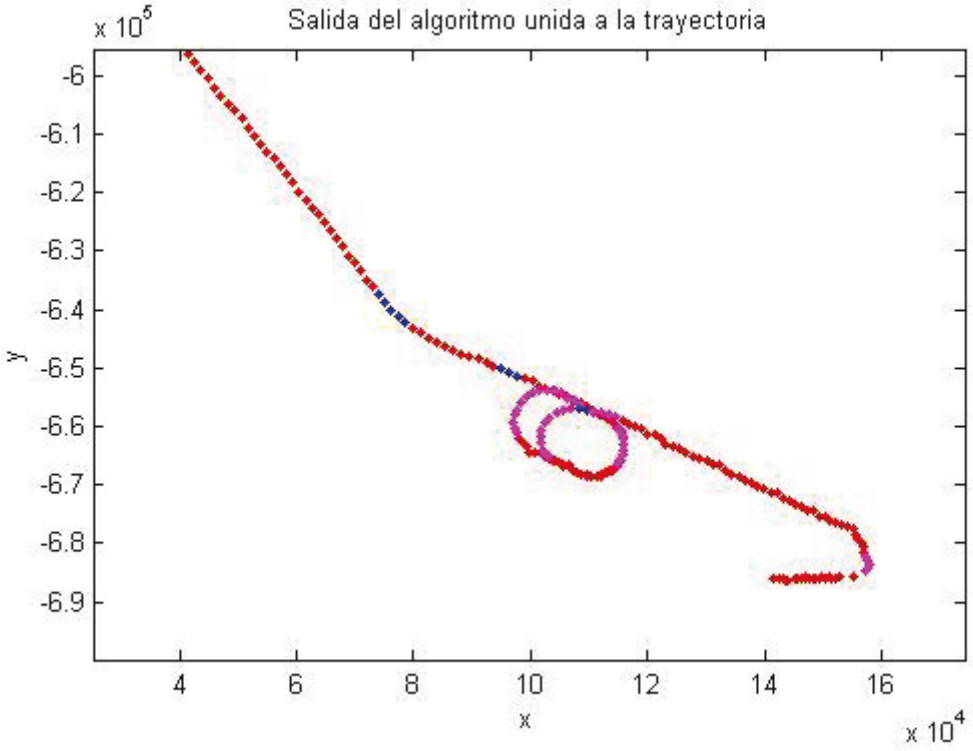


Figura 53G

Al final de la trayectoria, cuando las curvas son más pronunciadas, las distingue bastante bien, observando un giro a izquierdas, luego las dos circunferencias, y un último giro a derechas, aunque no lo consigue completo.

Trayectoria 2

En esta trayectoria, vemos que empieza por una gran recta, y posteriormente realiza un hipódromo, para después realizar una curva a izquierdas seguida de una curva a derechas.

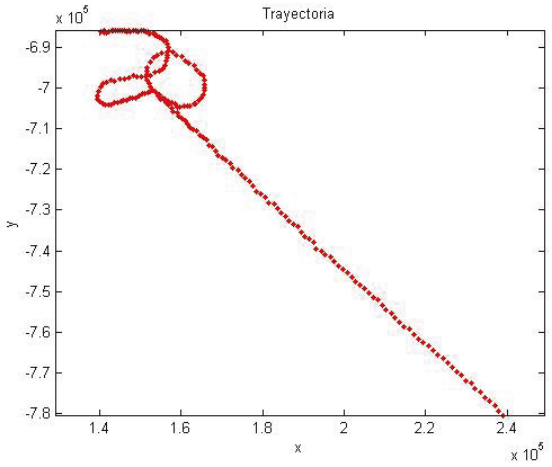


Figura 8B

La trayectoria descrita anteriormente, la podemos ver en el gráfico que nos muestra el tamaño de los ángulos.

Como tiene ese descenso en el tamaño que da 360 grados (circunferencia), y luego la curva formada por los giros seguidos.

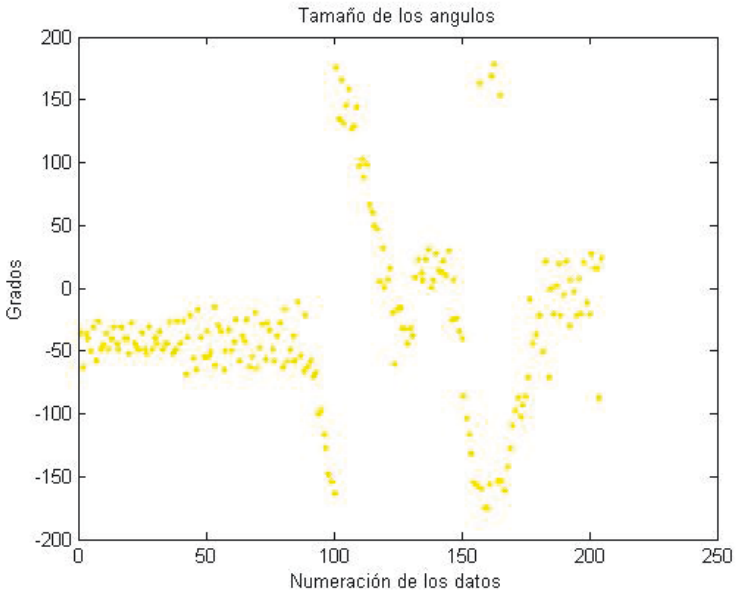


Figura 54A

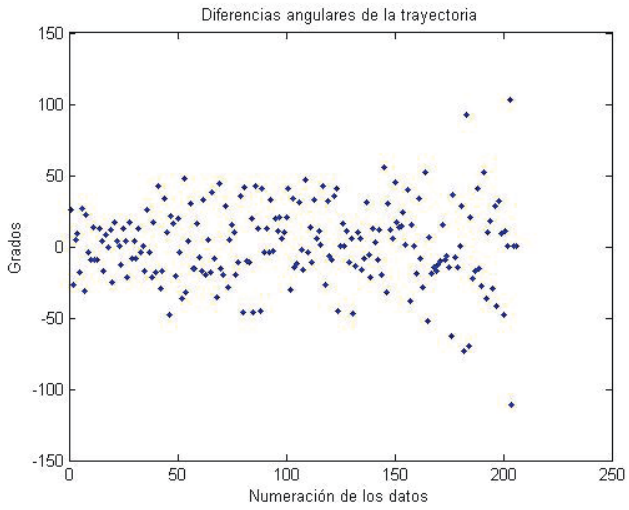


Figura 54B

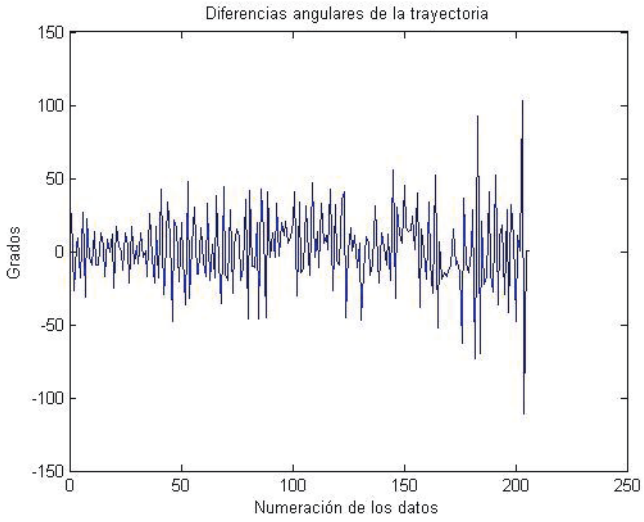


Figura 54C

El ruido dificulta la observación de las curvas en las diferencias angulares. No obstante, a partir del dato 100 podemos ver cómo hay sesgos que indican las curvas.

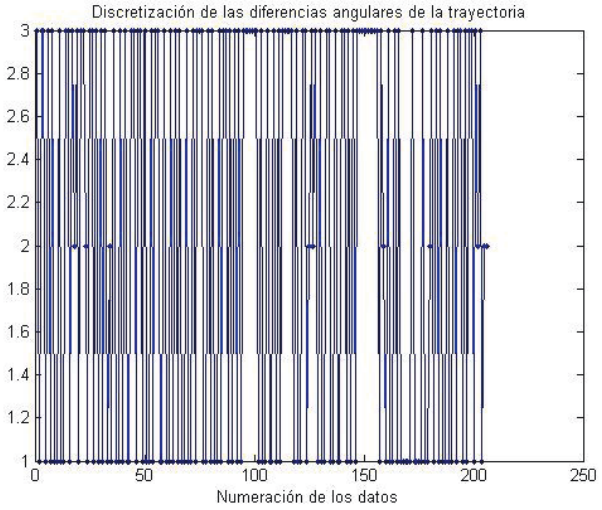


Figura 54D

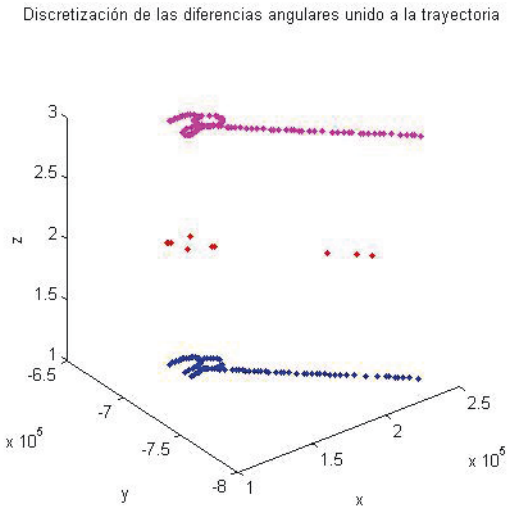


Figura 54E

Según se realiza la discretización de los datos, observamos que debido al ruido no discretiza al punto dos, que son las diferencias angulares más pequeñas que indican el estado de recto, por lo que si la salida del modelo fuera la misma, no sería un buen resultado.

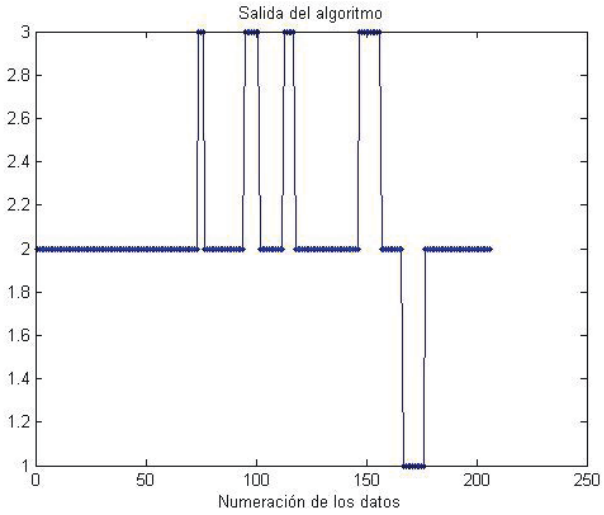


Figura 54F

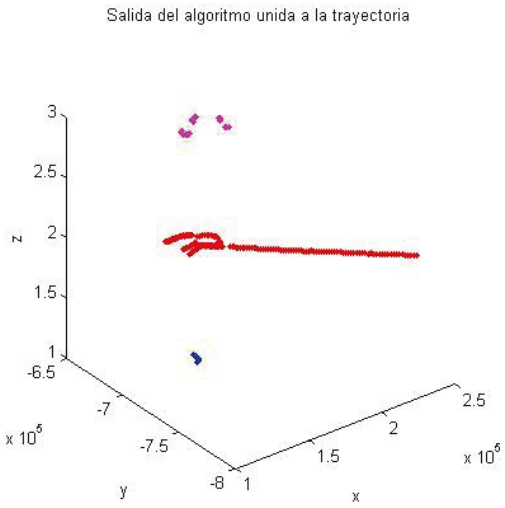


Figura 54G

Vemos cómo la salida del algoritmo es una clasificación más real. La recta principal la clasifica correctamente, excepto al final, que toma una curva a derechas. No obstante, clasifica el resto de la trayectoria correctamente, salvo una pequeña curva debido al ruido. Omitiendo esa excepción, el resto de la trayectoria la clasifica perfectamente.

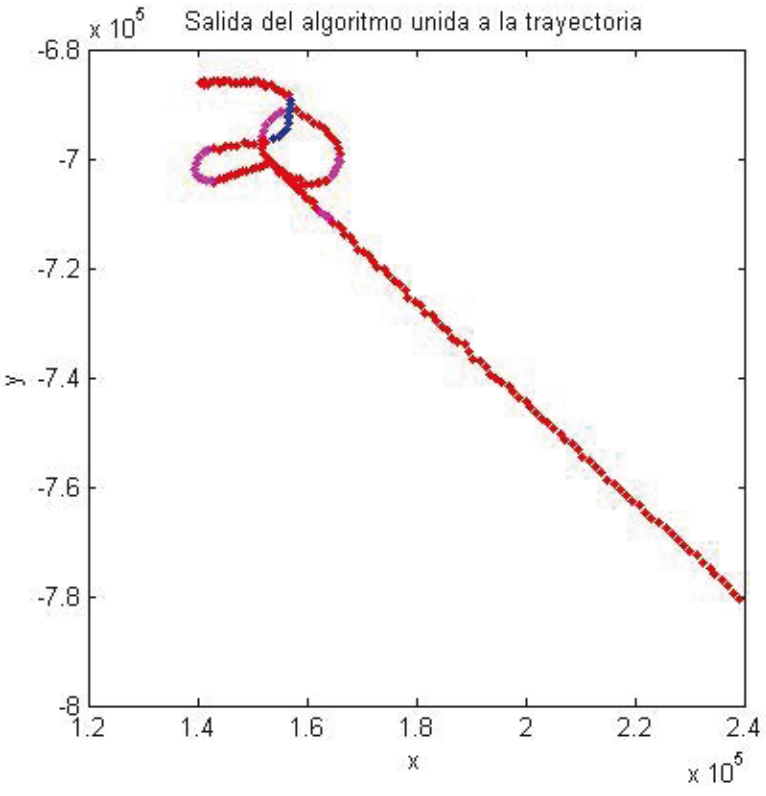


Figura 54I

Trayectoria 3

La trayectoria 3 es muy parecida al anterior por lo que podremos comparar los resultados y poder realizar alguna conclusión sobre la naturaleza de los fallos.

Prácticamente la única diferencia es aparte del grado de las curvas, la falta de una curva después del hipódromo. También vemos que en realidad no son iguales, sino simétricas, ya que las curvas tomadas a izquierdas, ahora son a derechas y viceversa.

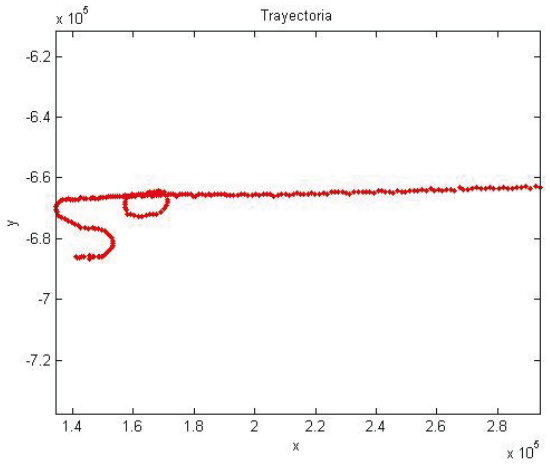


Figura 8C

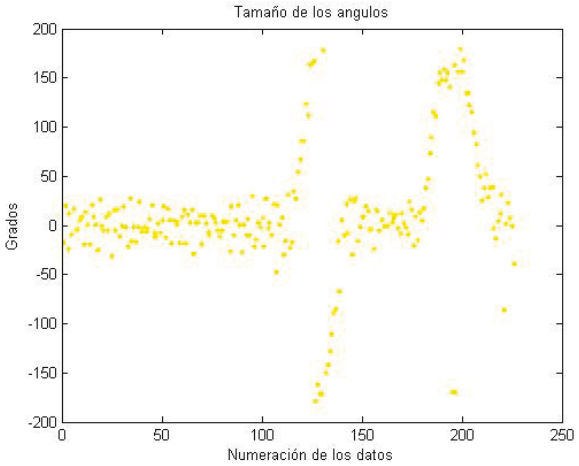


Figura 55A

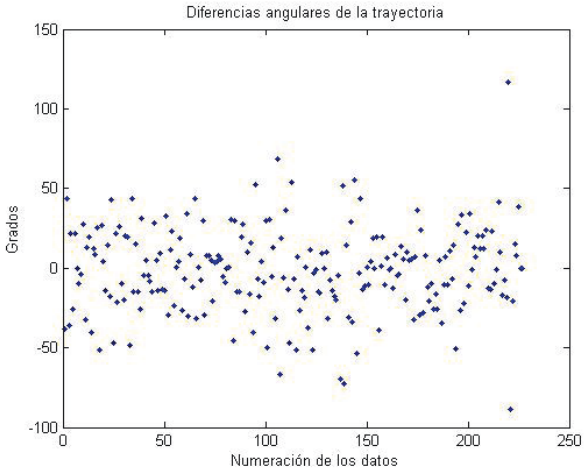


Figura 55B

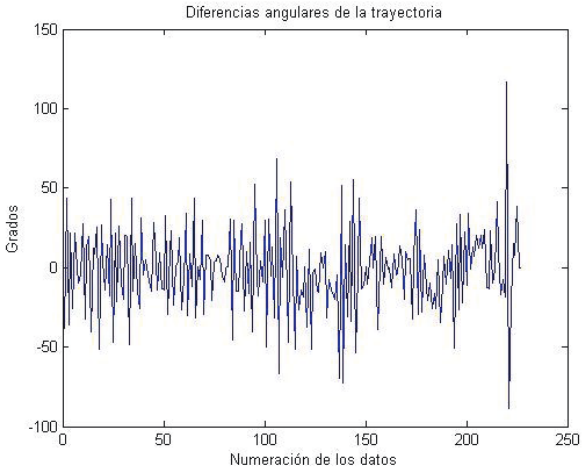


Figura 55C

Como podemos ver, los gráficos apoyan la idea de que las trayectorias son muy similares, salvando las distintas distancias a los sensores de cada una en cada punto podemos ver cómo tiene las mismas trazas las dos trayectorias.

Ya hemos visto este problema antes, al discretizar las trayectorias no discretiza a los estados donde se supone que deberían estar las rectas por el ruido, pero sabemos que el algoritmo junto con el modelo podrá solucionarlo cuando le introduzcamos los observables.

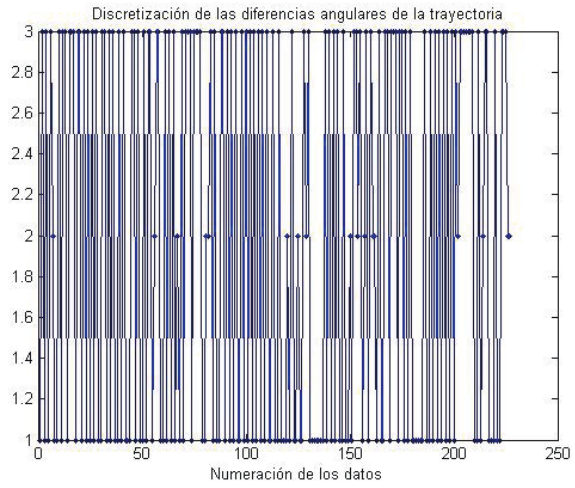


Figura 55D

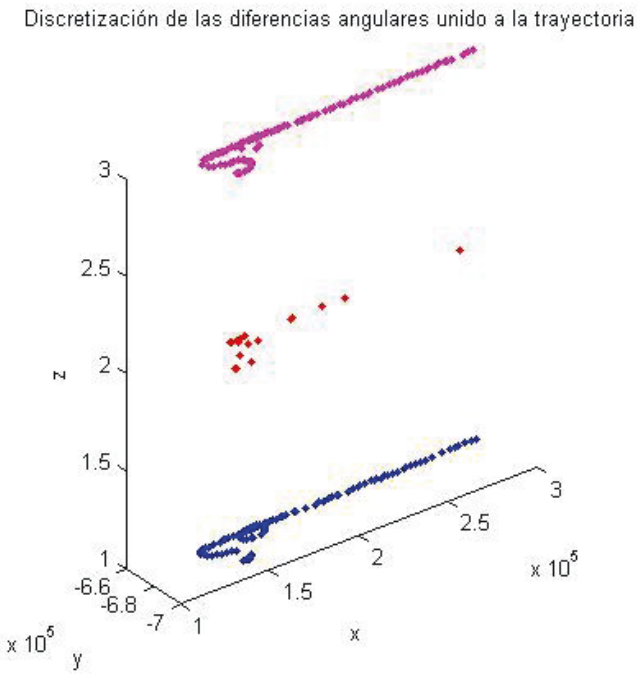


Figura 55E

Una vez tenemos los observables, se los pasamos al algoritmo y vemos la salida de éste:

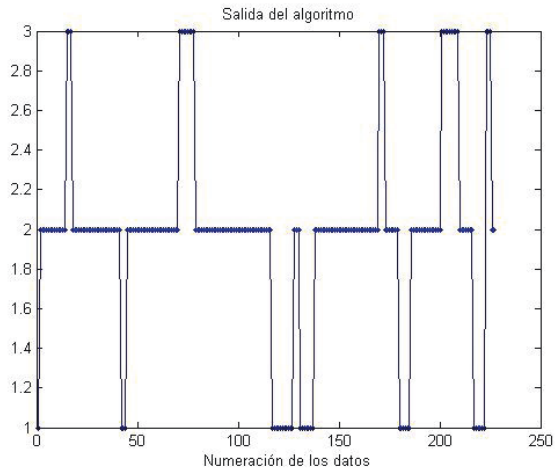


Figura 55F

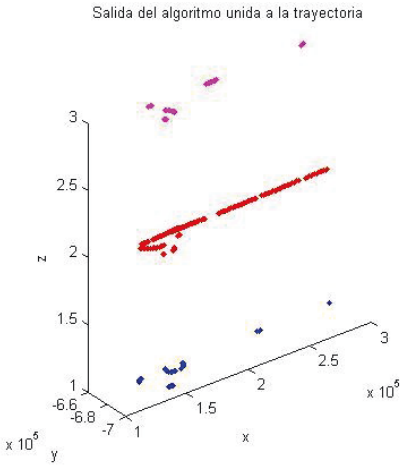


Figura 55G

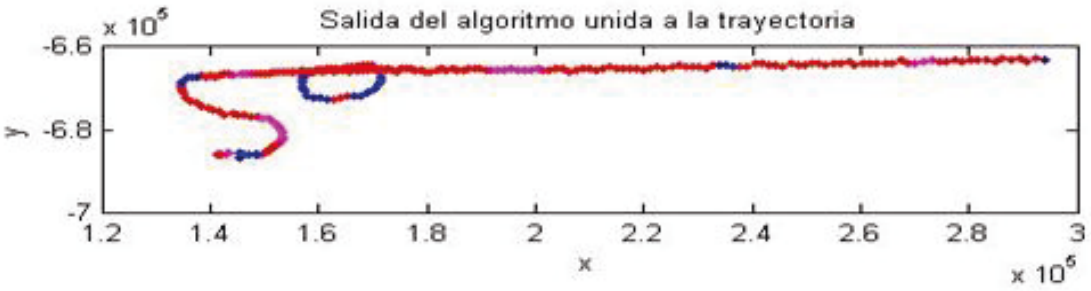


Figura 55H

En este caso, realiza una peor clasificación debido a factores como el ruido por la distancia al sensor. Le resulta más sencillo la clasificación de las curvas cuanto más cerradas sean, como en el caso de la trayectoria 2, que tenía curvas más cerradas. Por tanto, la clasificación en ésta es peor.

Trayectoria 4

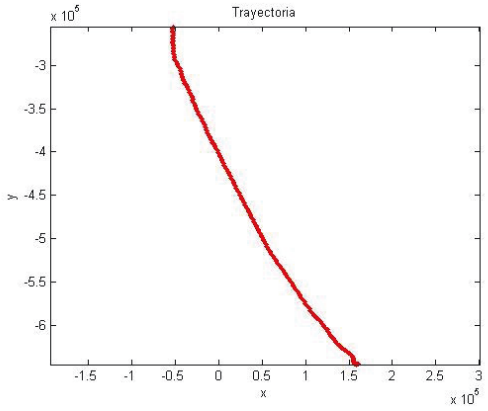


Figura 8D

Como vemos, la trayectoria es una ligera curva que finaliza con una curva más pronunciada en la parte inferior de la gráfica.

Los tamaños de los ángulos no ofrecen mucha información, ya que la curva es tan suave que el ruido prevalece por encima. Por tanto, las zonas lejanas al radar son las que aparecen más dispersas por los distintos tamaños de los ángulos.

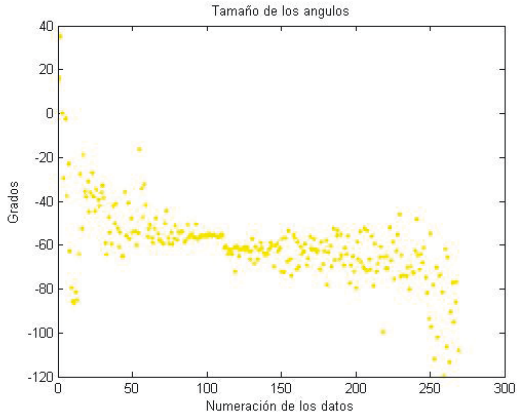


Figura 56A

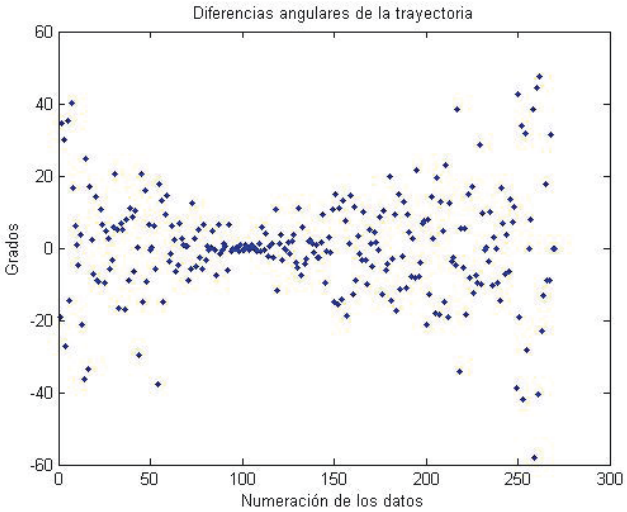


Figura 56B

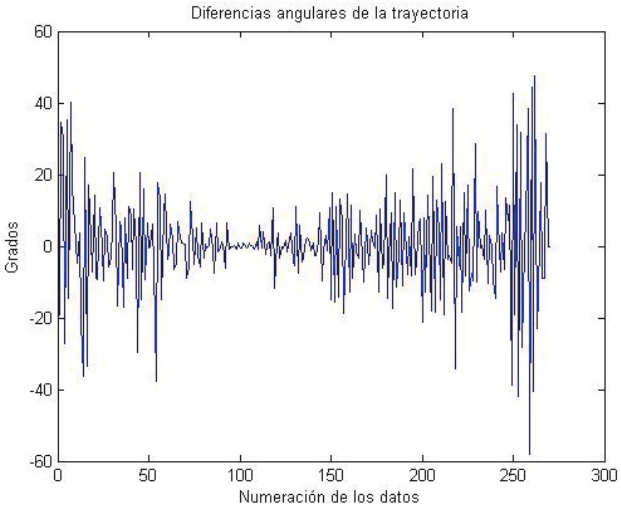


Figura 56C

Como vemos en los gráficos, las diferencias angulares no parecen representar adecuadamente la trayectoria, pero esto ya ha sucedido anteriormente, y el algoritmo ha conseguido resolver el problema.

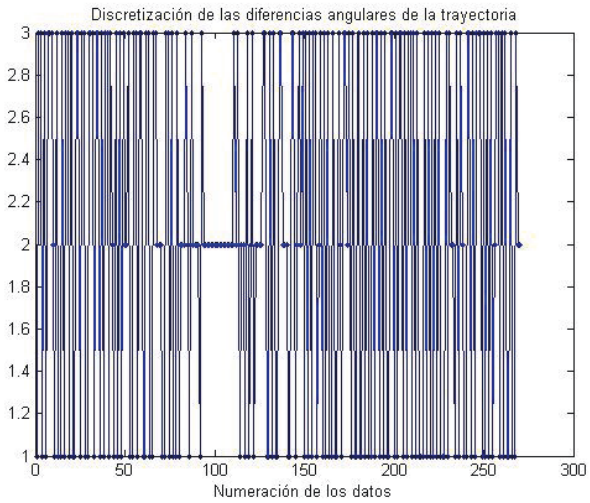


Figura 56D

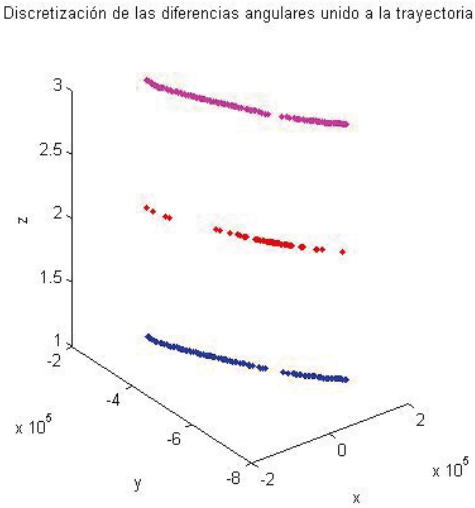


Figura 56E

Una vez discretizamos y conseguimos los observables, se los pasamos al algoritmo para que nos muestre la salida.

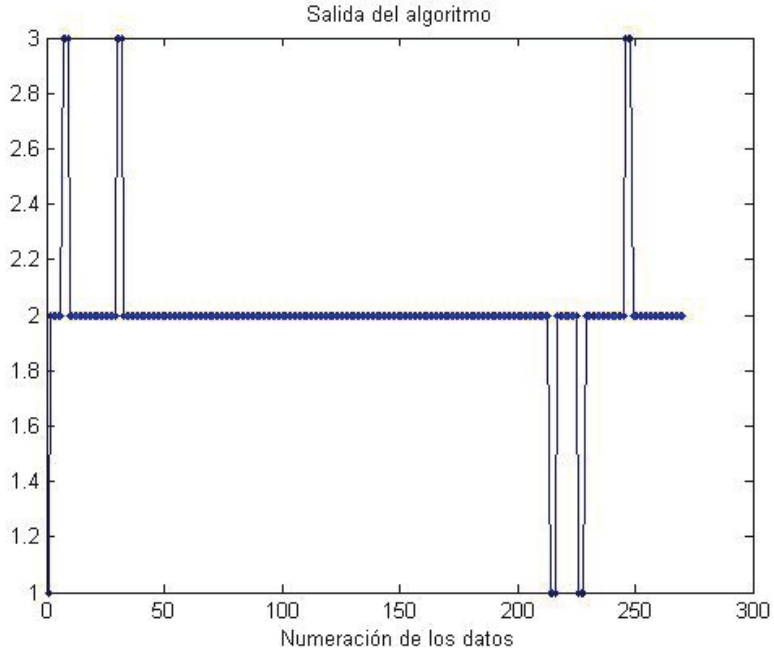


Figura 56F

Vamos a unir la salida a la trayectoria, pero por lo que vemos, posiblemente no sea el resultado esperado.

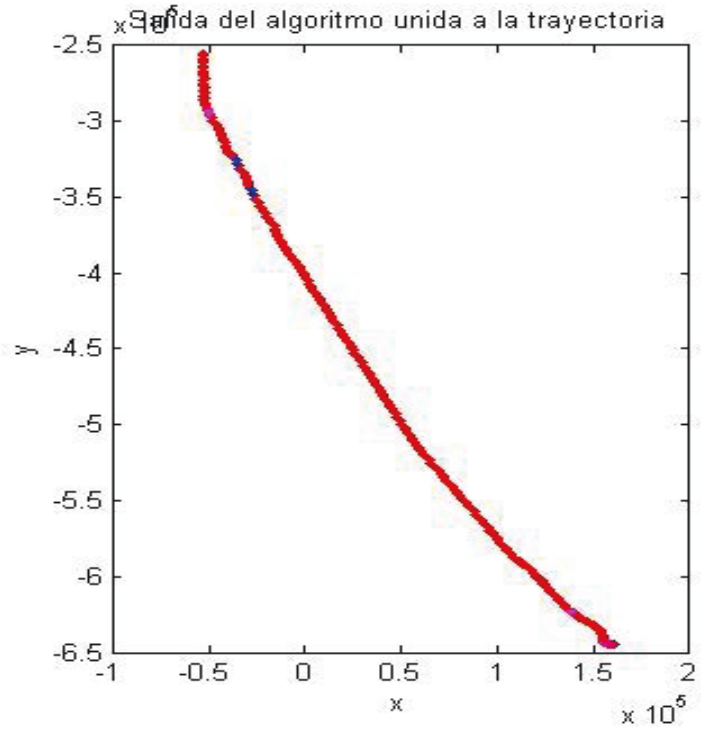


Figura 56G

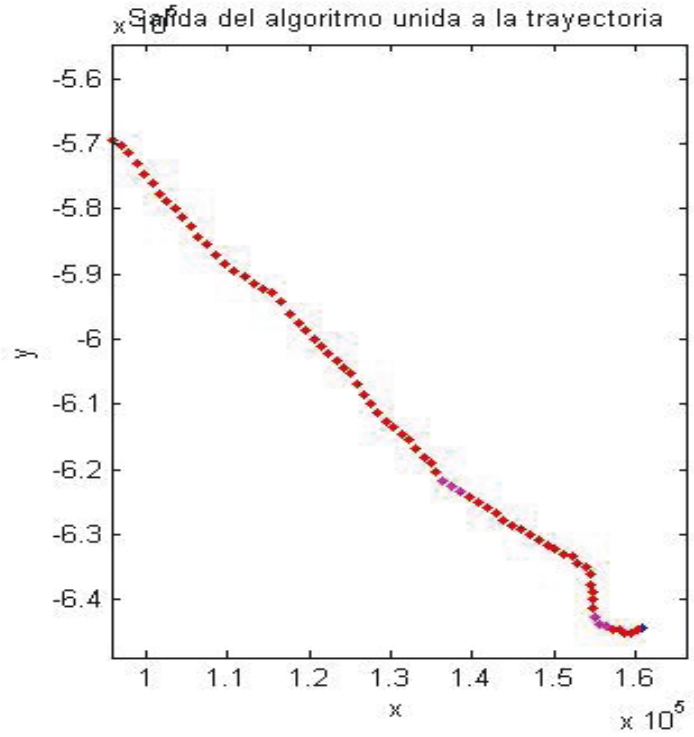


Figura 56H

Hemos visto que toda la curva la clasifica como una recta, lo cual era de esperar pero centrándonos en la última curva más pronunciada, vemos que no realiza una correcta clasificación. A continuación veremos si aplicando el filtro de Kalman a los datos, podemos obtener mejores resultados.

14.2. Trayectorias reales después de aplicar el filtro de Kalman

El modelo usado para las trayectorias a las que se les aplicado el filtro de Kalman es el siguiente:

Estado Siguiente / Estado Actual	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Giro Izq. (1)	0.65	0.25	0.1
Recto (2)	0.1	0.8	0.1
Giro Dcha. (3)	0.1	0.25	0.65

Tabla 90: Matriz de transición del modelo para datos reales con filtro de Kalman

Acción / Observación	Giro Izq. (1)	Recto (2)	Giro Dcha. (3)
Diferencia angular grande negativa (1)	0.55	0.4	0.05
Diferencia angular pequeña (2)	0.1	0.8	0.1
Diferencia angular grande positiva (3)	0.05	0.4	0.55

Tabla 91: Matriz de observación del modelo para datos reales con filtro de Kalman

Los topes usados en la discretización de los datos son 3 y -3.

Trayectoria 1

Recordamos que el filtro de Kalman suaviza el efecto del ruido, por lo que la trayectoria será más continua sin los saltos producidos por el ruido. Además, cuando el ruido es muy elevado retrasa las curvas, y las curvas más ligeras las elimina, ya que las puede confundir con ruido. Por lo que no son los datos exactos tomados por los sensores.

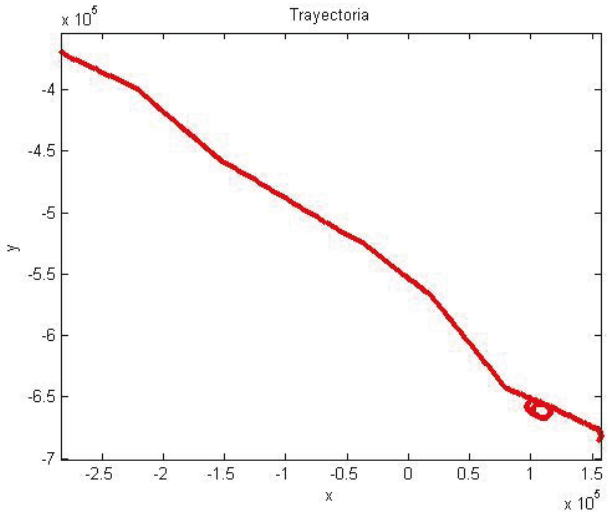


Figura 57A

Vemos que los tamaños de los ángulos son similares mientras la trayectoria se mantiene recta, lo que muestra el efecto del filtro, ya que antes tenían variaciones mucho mayores.

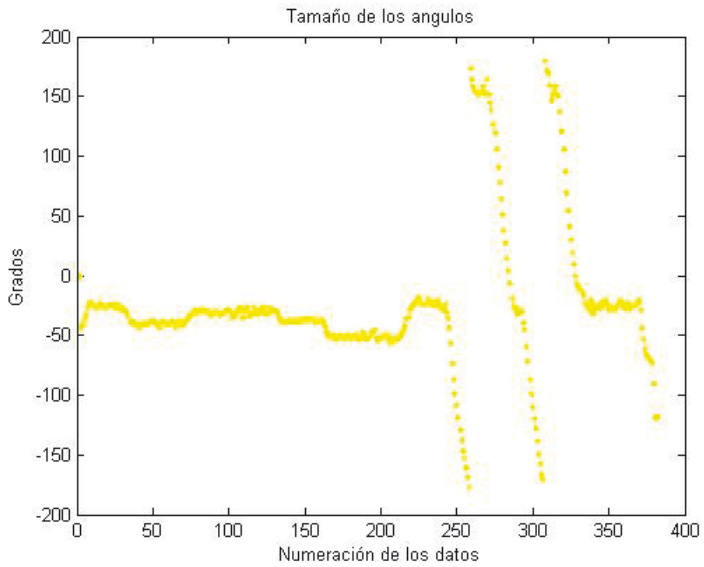


Figura 57B

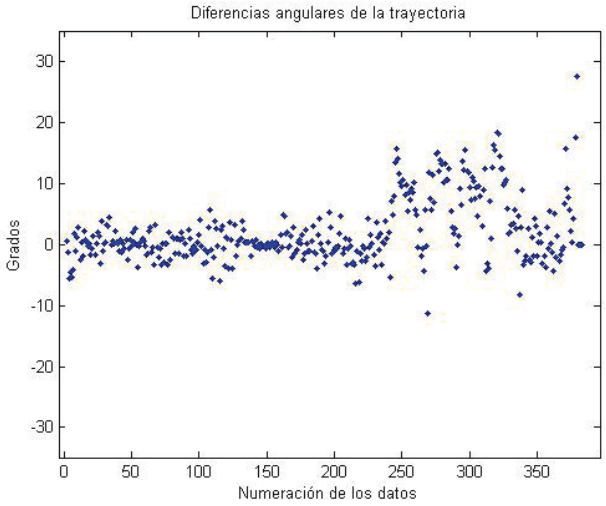


Figura 57C

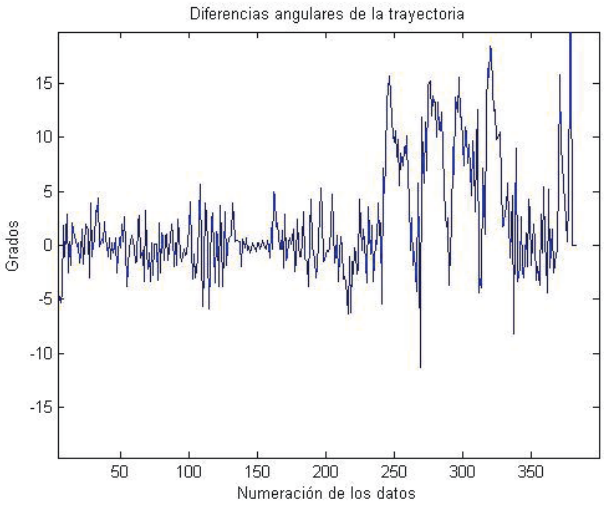


Figura 57D

Se ven claramente las curvas más pronunciadas al final de la trayectoria, pero las que son ligeras se pierden por el ruido y el filtro aplicado.

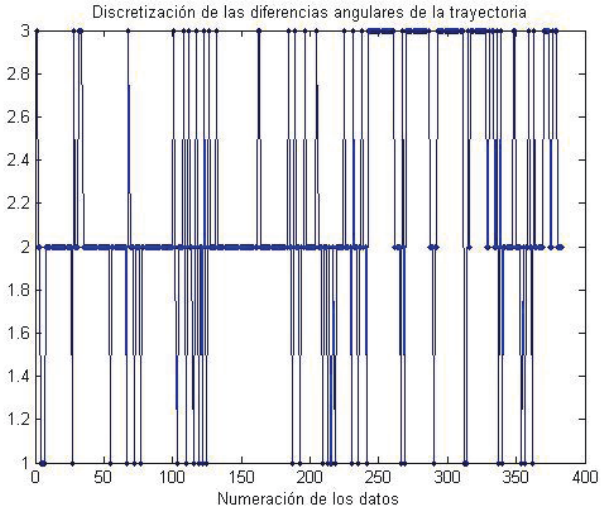


Figura 57E

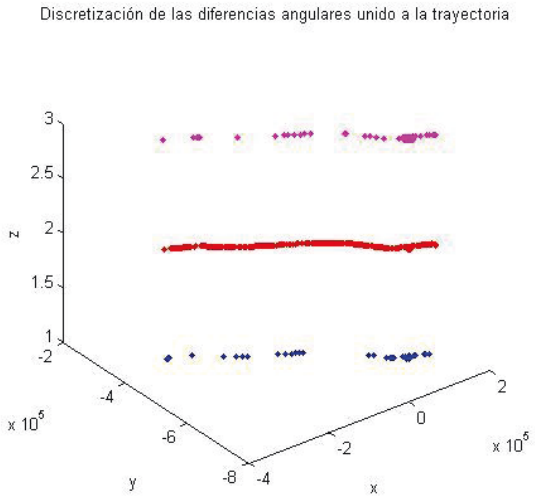


Figura 57F

En la discretización ya vemos que los observables tienen mejor aspecto, ya que la mayor parte de la trayectoria es discretizada como 2, diferencia angular cercana a cero.

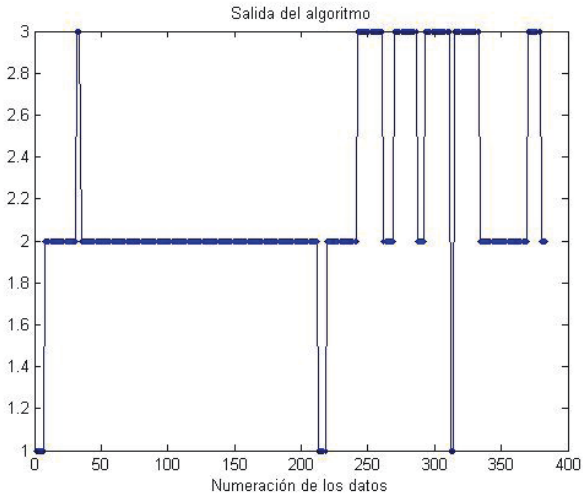


Figura 57G

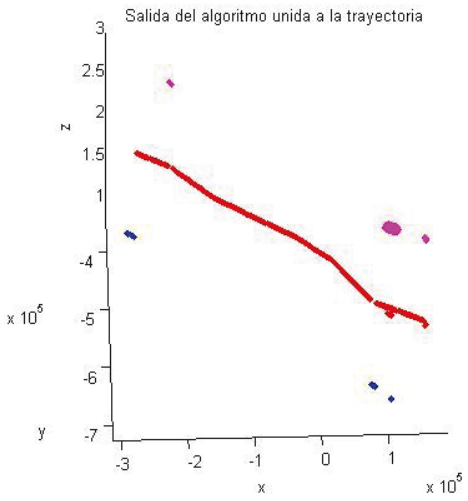


Figura 57H

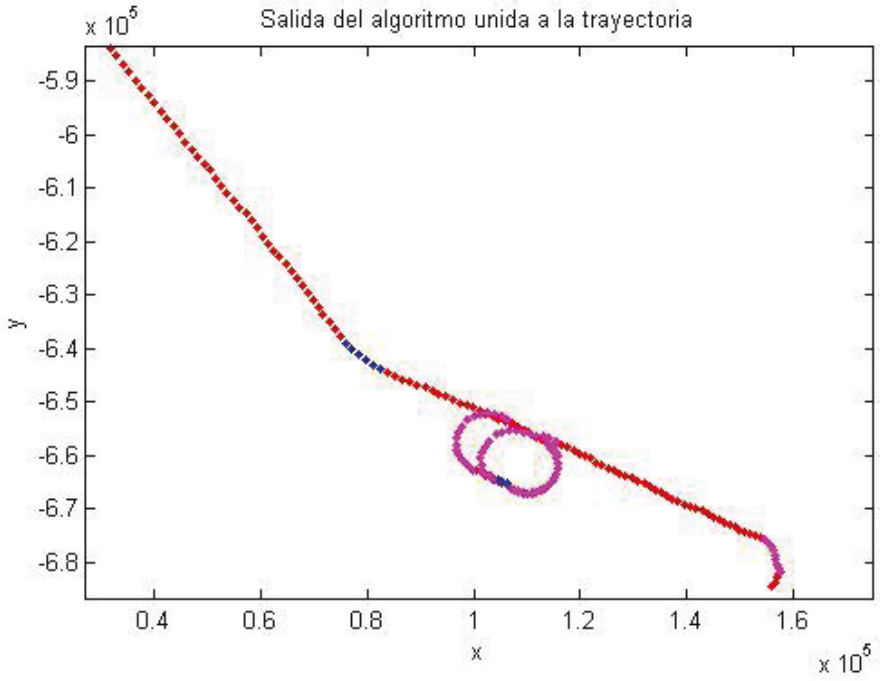


Figura 57I

Vemos en los gráficos cómo consigue clasificar correctamente pequeñas curvas que tiene la trayectoria en sus inicios, pero no lo consigue con todas. No obstante, si nos centramos en el final de la trayectoria donde las curvas son más pronunciadas, vemos cómo las clasifica casi perfectamente. Todas las curvas son reconocidas, aunque puede que pierdan o ganen más tamaño del real.

Trayectoria 2

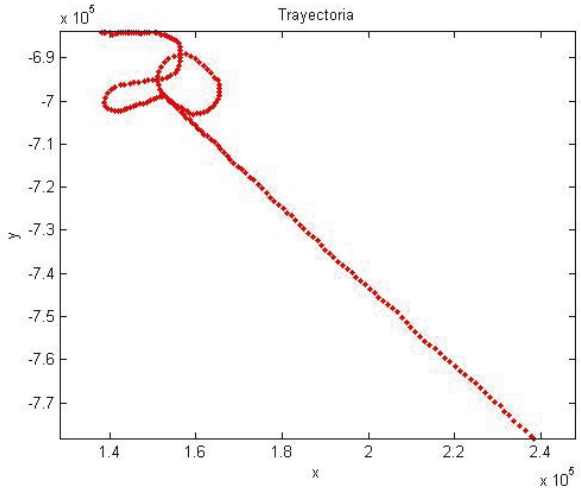


Figura 58A

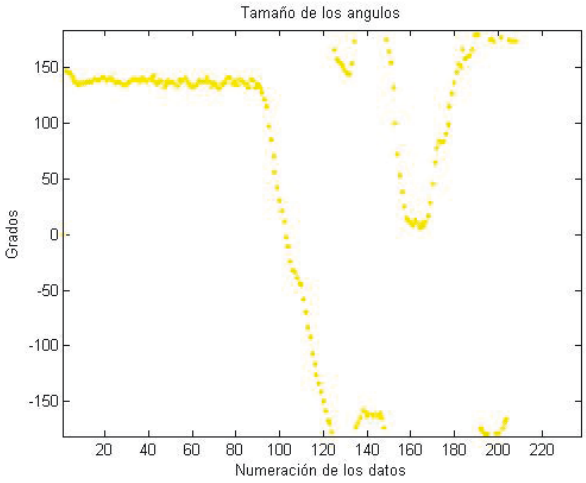


Figura 58B

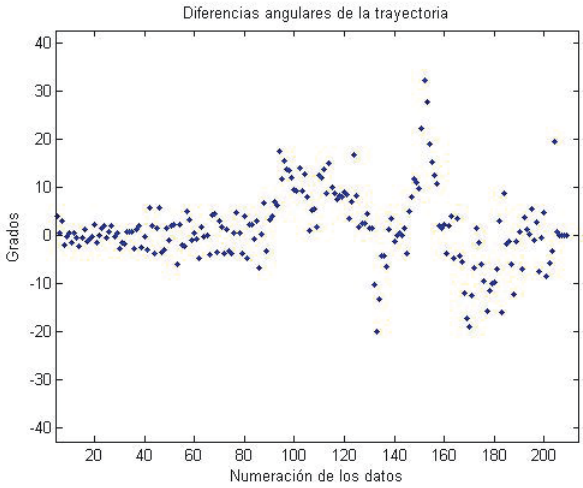


Figura 58C

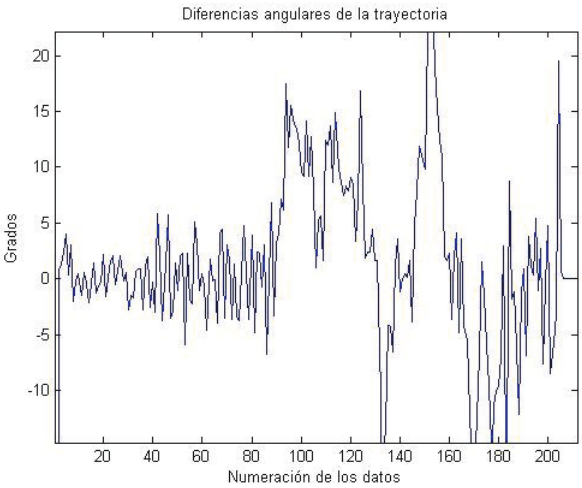


Figura 58D

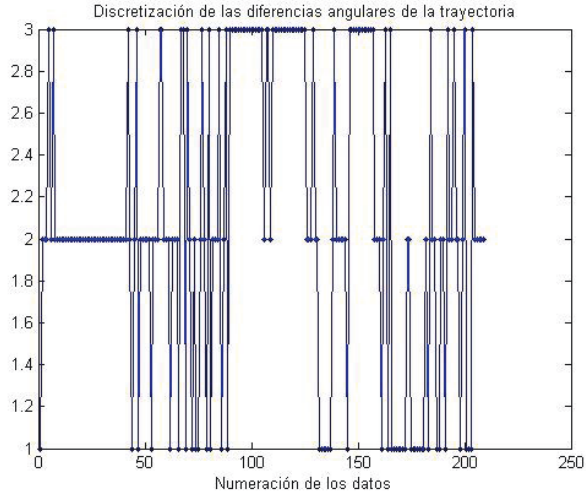


Figura 58E

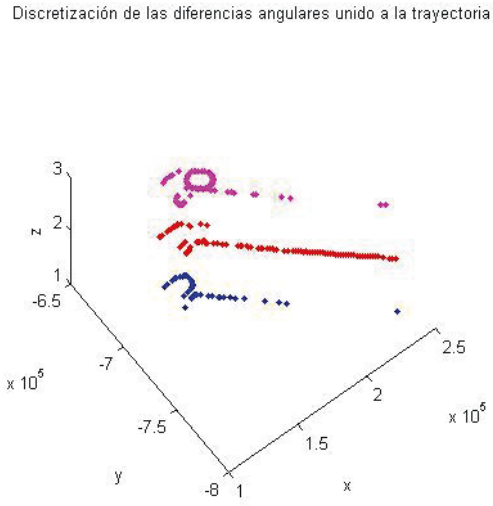


Figura 58F

Viendo los datos de entrada y los cambios realizados para convertirlos en los observables que pasaremos al algoritmo para que obtengamos la salida, podemos deducir que la clasificación va a ser buena ya que son datos claros y definidos, lo que permitirá que el algoritmo pueda trabajar junto al modelo de forma sencilla y confiar en los observables para conseguir una buena clasificación.

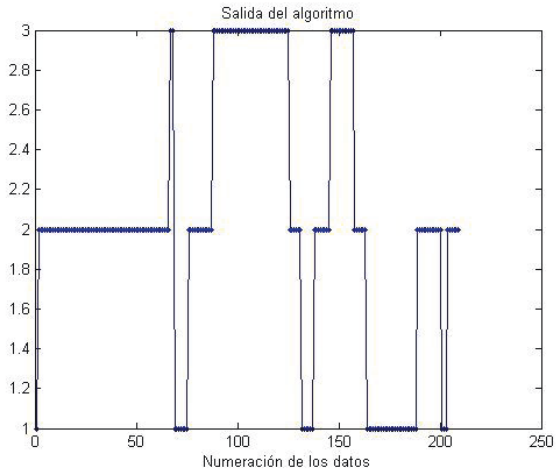


Figura 58G

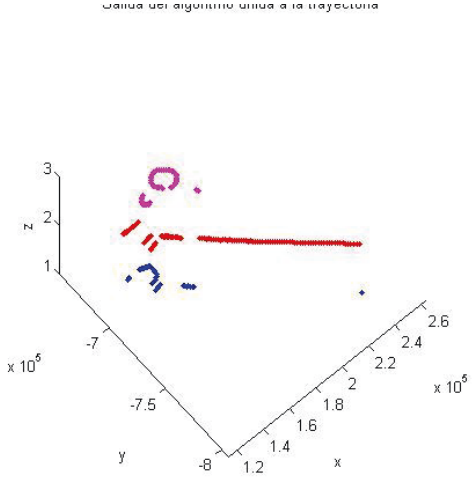


Figura 58H

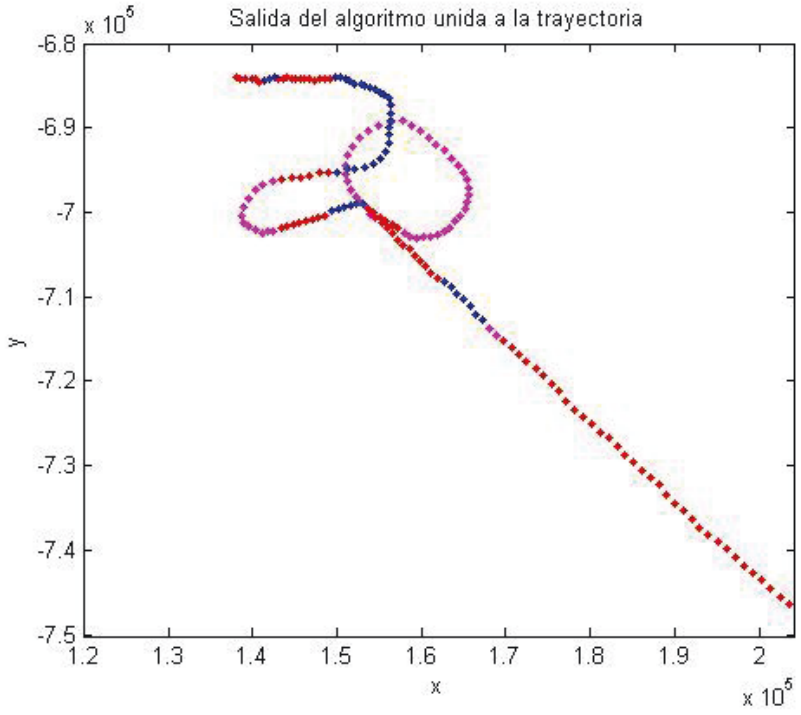


Figura 58I

Vemos que es un resultado prometedor, tiene algunos fallos pero aun así consigue un mejor resultado en comparación a la trayectoria sin aplicar el filtro.

Trayectoria 3

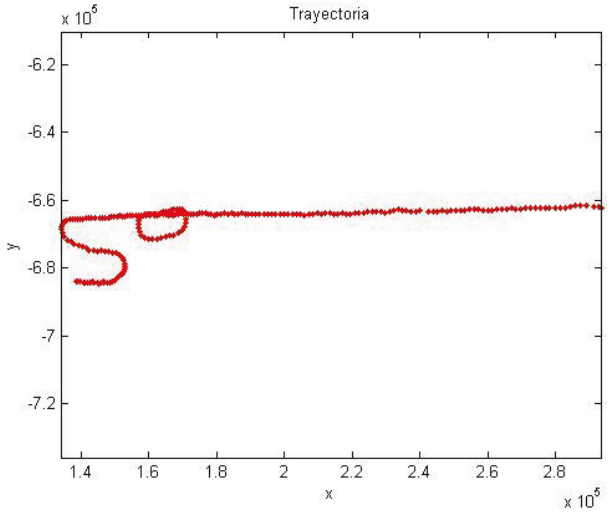


Figura 59A

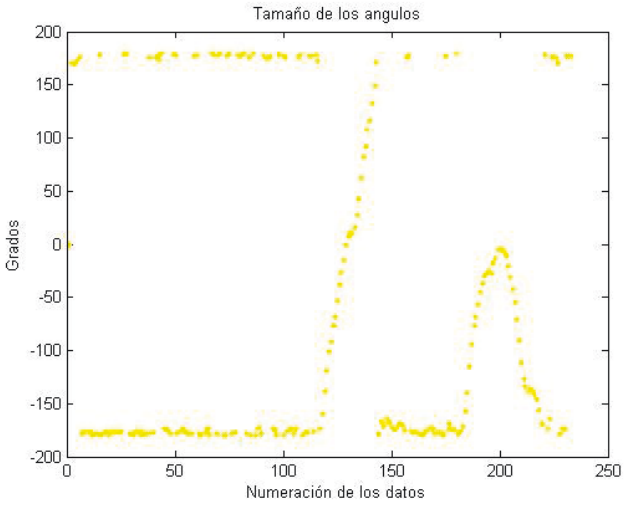


Figura 59B

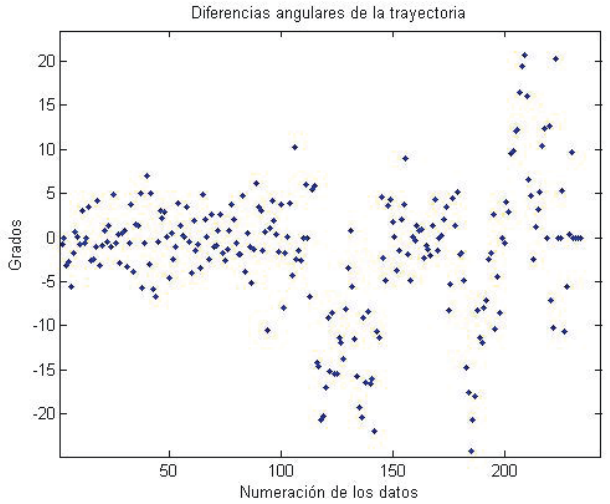


Figura 59C

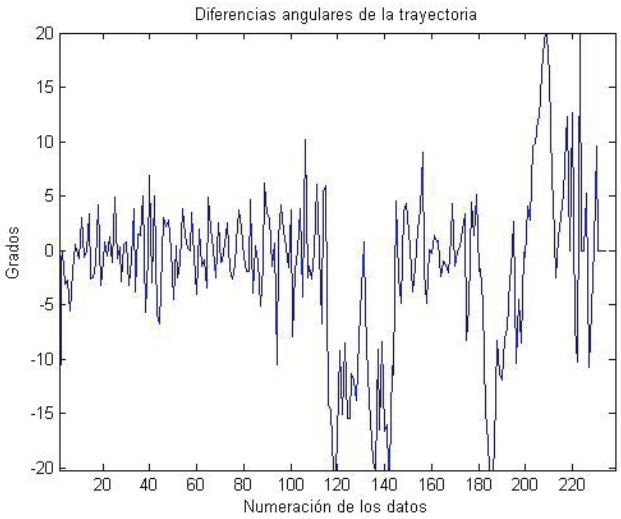


Figura 59D

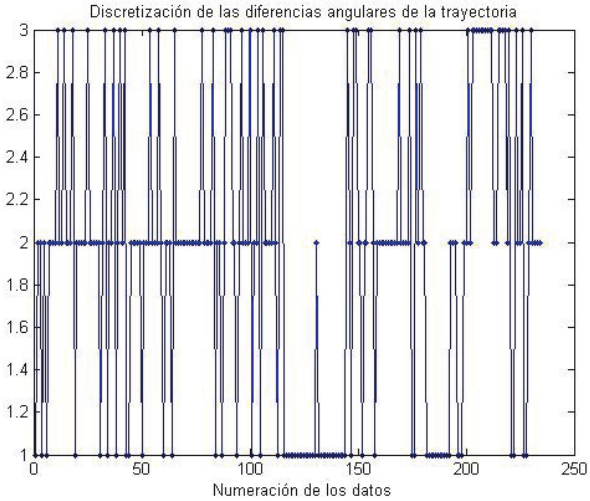


Figura 59E

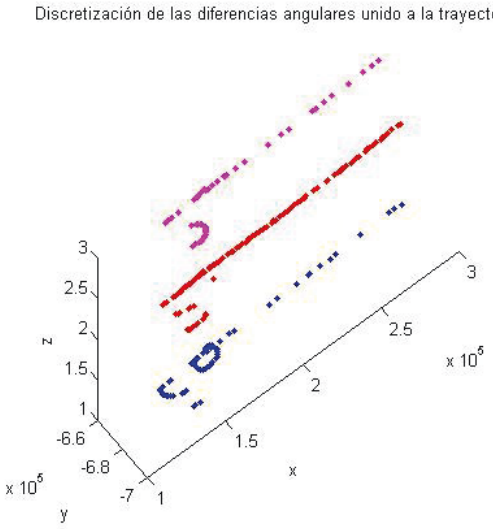


Figura 59F

Podemos ver una irregularidad en el tamaño de los ángulos de la trayectoria, totalmente explicable. En la trayectoria puede obtener un ángulo de 360 grados, pero por las funciones usadas para obtener los ángulos el tamaño funciona de 180 a -180, es decir, si partiéramos de 0 a 180 y a 360 sería lo mismo que partir de -180 a 0 y a 180, simplemente cambia la manera de representarlo. Que los tamaños vayan de 180 a -180 se debe al salto que daría de ir de 360 a 0, son ángulos que están al lado pero en la gráfica los veríamos igualmente separados, es lo mismo que sucede en este caso.

Por lo demás, es muy similar a la tercera trayectoria y los datos son bastante claros, por lo que esperamos obtener un buen resultado.

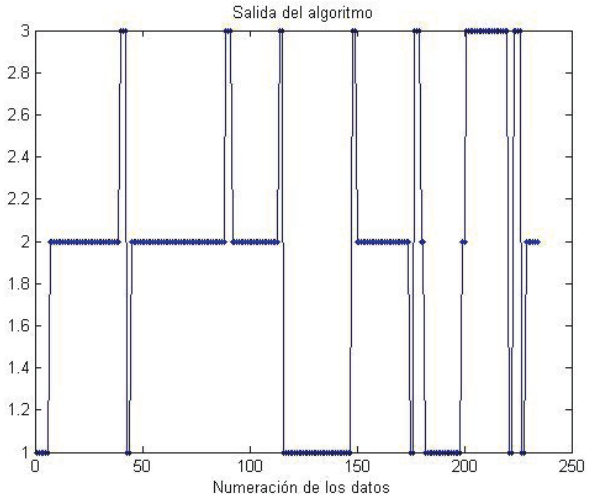


Figura 59G

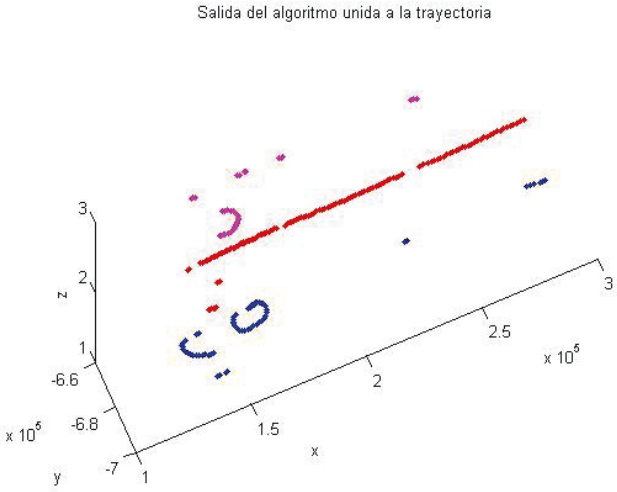


Figura 59H

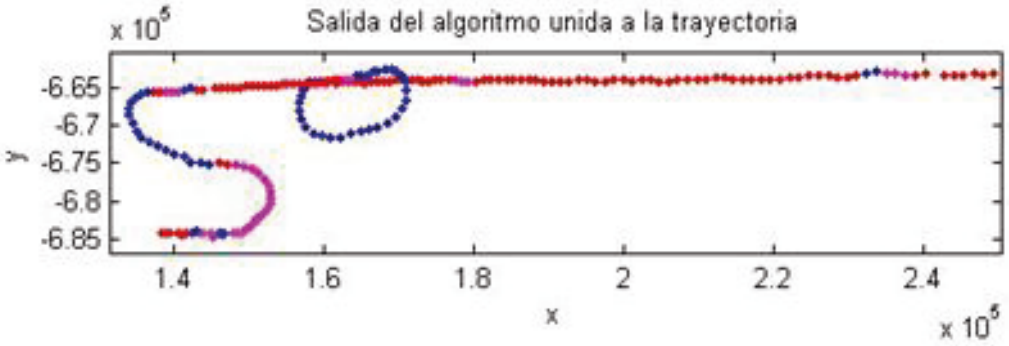


Figura 59I

Como esperábamos, el resultado es bastante bueno. Vemos cómo en el hipódromo se ha producido una ligera deformación al aplicar el filtro de Kalman, pero eso no es problema del clasificador, ya que el sólo puede clasificar en función de los observables. Omitiendo este pequeño detalle y algunas rectas mal clasificadas, todo lo restante no contiene errores.

Trayectoria 4

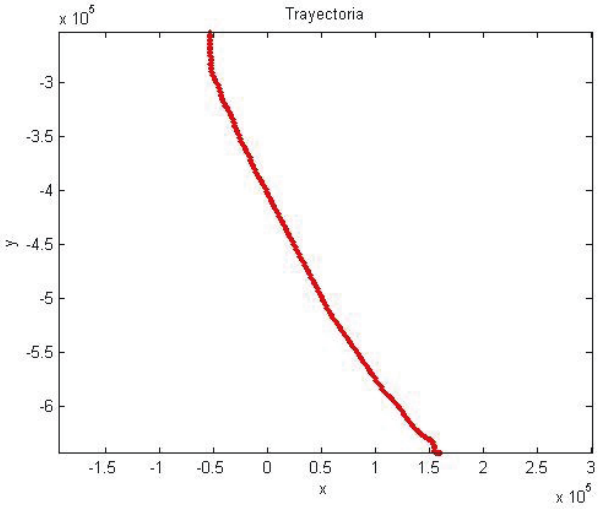


Figura 60A

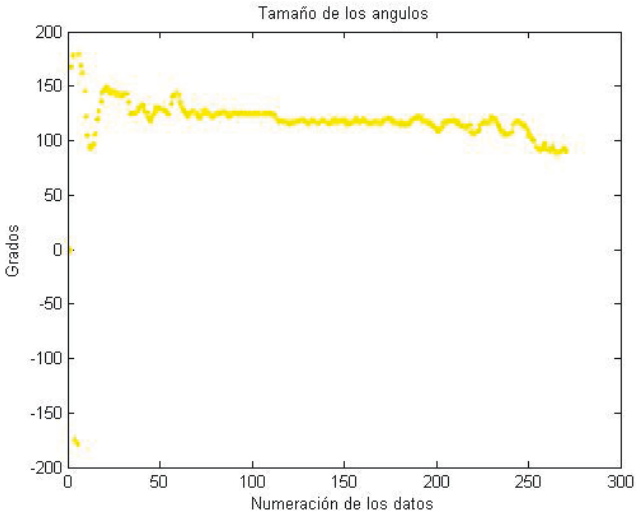


Figura 60B

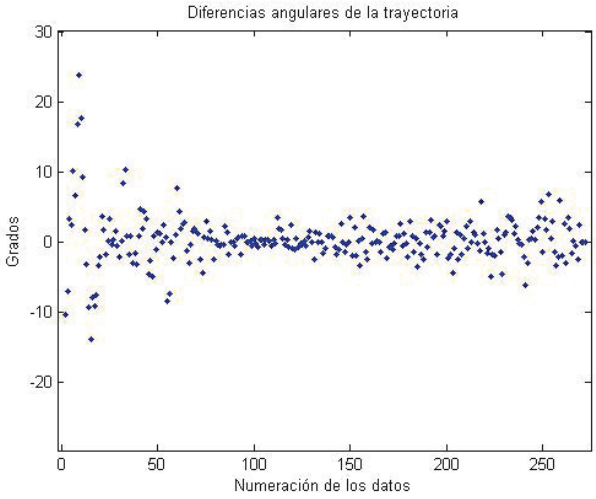


Figura 60C

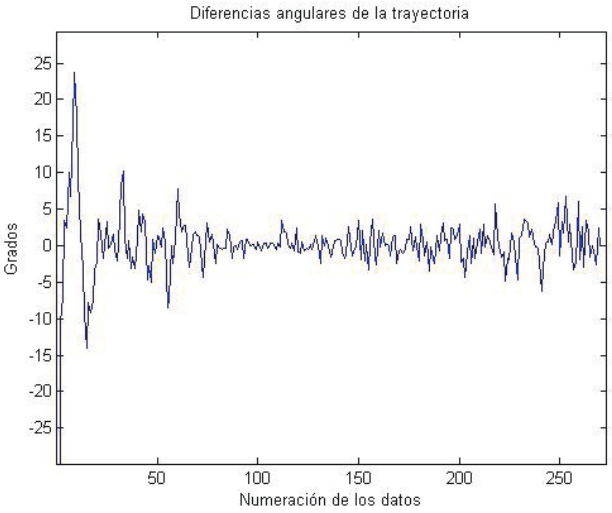


Figura 60D

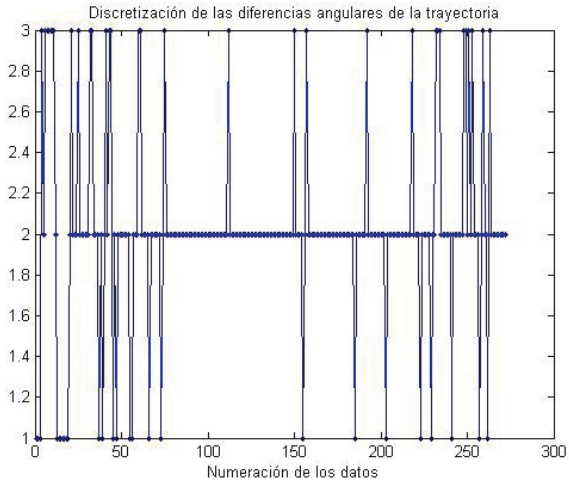


Figura 60E

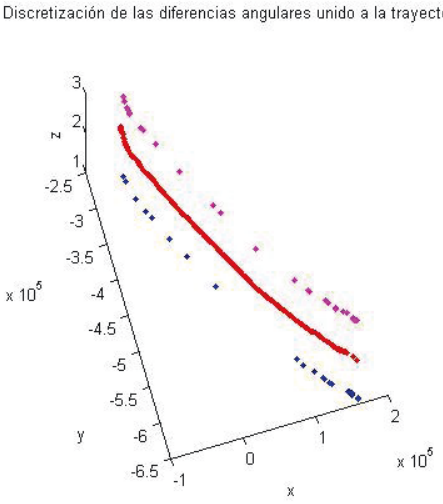


Figura 60F

Estamos viendo que la trayectoria no tiene grandes cambios de dirección, las curvas son casi inapreciables, y además vamos a ver cómo al inicio de la trayectoria por el ruido el filtro de Kalman tarda en estabilizarse, lo que provocará que el algoritmo falle al clasificar el principio de la trayectoria.

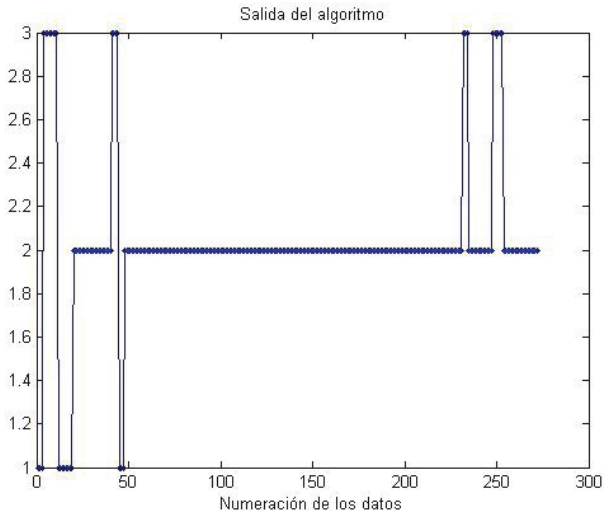


Figura 60G

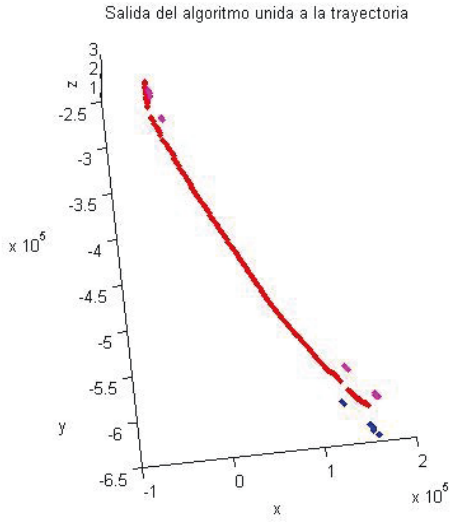


Figura 60H

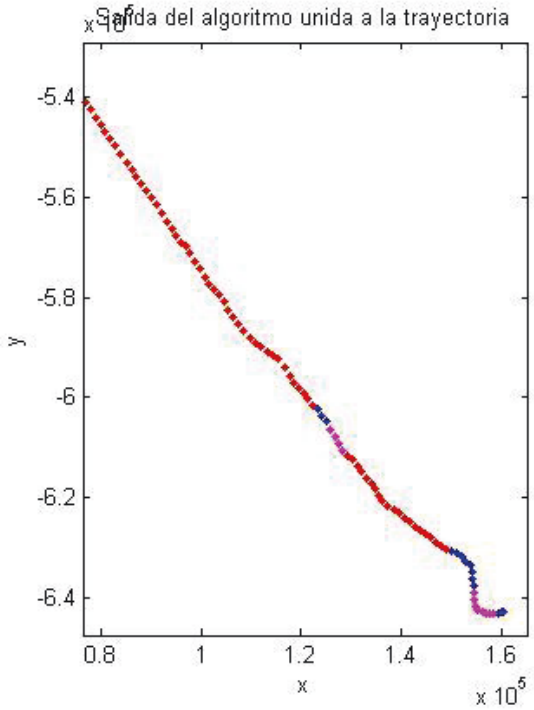


Figura 60I

Como ya sabíamos, la trayectoria entera no la toma como una curva, ya que está pensado para curvas más pronunciadas y pequeñas, y la toma como una recta en la mayor parte del tiempo excepto al principio, que comete un ligero error. También hay una curva ligeramente más pronunciada. No obstante, si nos centramos al final de ésta vemos cómo tiene algunas curvas que clasifica correctamente.

14.3. Comparación de trayectorias reales

Trayectoria 1

Al no tener la trayectoria ideal, no podemos saber el porcentaje de acierto exacto pero podemos observar gráficamente los errores y los cambios. Mayormente nos centraremos en los cambios que tenemos entre las distintas trayectorias, y en qué casos mejora la clasificación aplicar el filtro de Kalman.

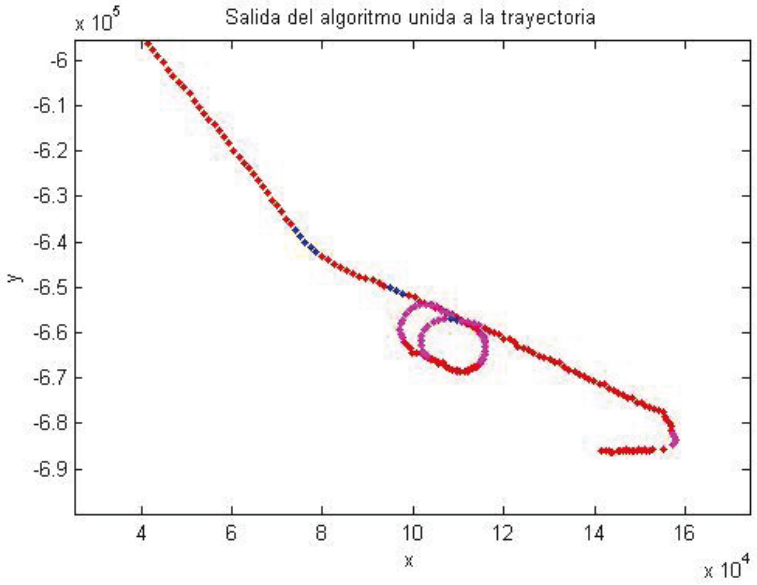


Figura 61: Trayectoria 1 sin filtro de Kalman

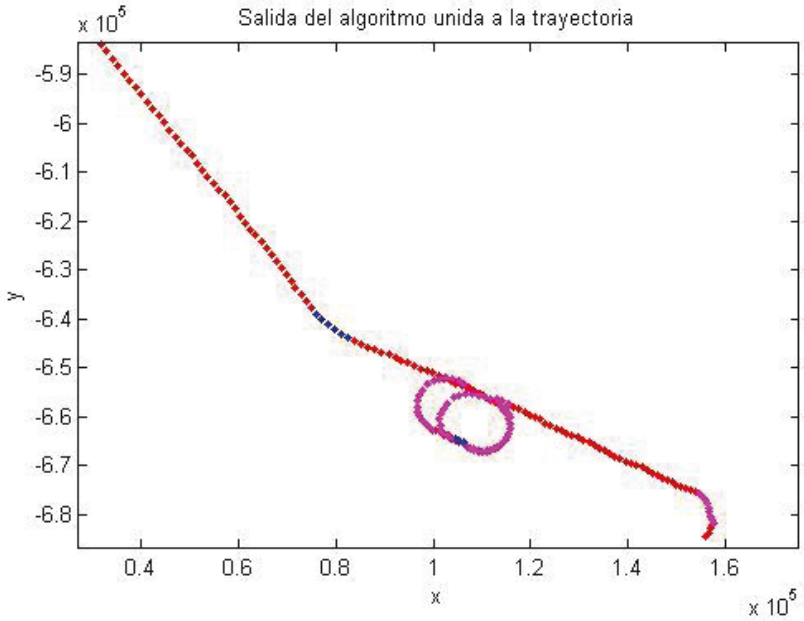


Figura 62: Trayectoria 1 con filtro de Kalman

Como hemos podido ver, el filtro de Kalman llega a modificar la trayectoria, al igual que lo hace el ruido. No obstante, podemos suponer que ninguna de las dos graficas nos muestra la trayectoria exactamente por los diversos factores.

Vemos que la que no tiene filtro de Kalman realiza una buena clasificación hasta la última curva, en la que no reconoce la mitad de la curva, sólo una pequeña parte, y con el filtro de Kalman la curva es mucho más pequeña, y consigue una clasificación buena.

No obstante, en los hipódromos clasifica el hipódromo entero como curva, cosa que no debería darse, ya que como aprendimos de la primera trayectoria simulada no todo el hipódromo forma la curva.

Sin embargo, en la primera parte de la trayectoria cuando se le aplica el filtro, el algoritmo distingue y clasifica algunas curvas que, sin aplicar el filtro a los datos de entrada, no clasifica correctamente.

Por tanto, creo que tiene un mayor número de aciertos en la clasificación una vez aplicado el filtro de Kalman a la trayectoria.

Trayectoria 2

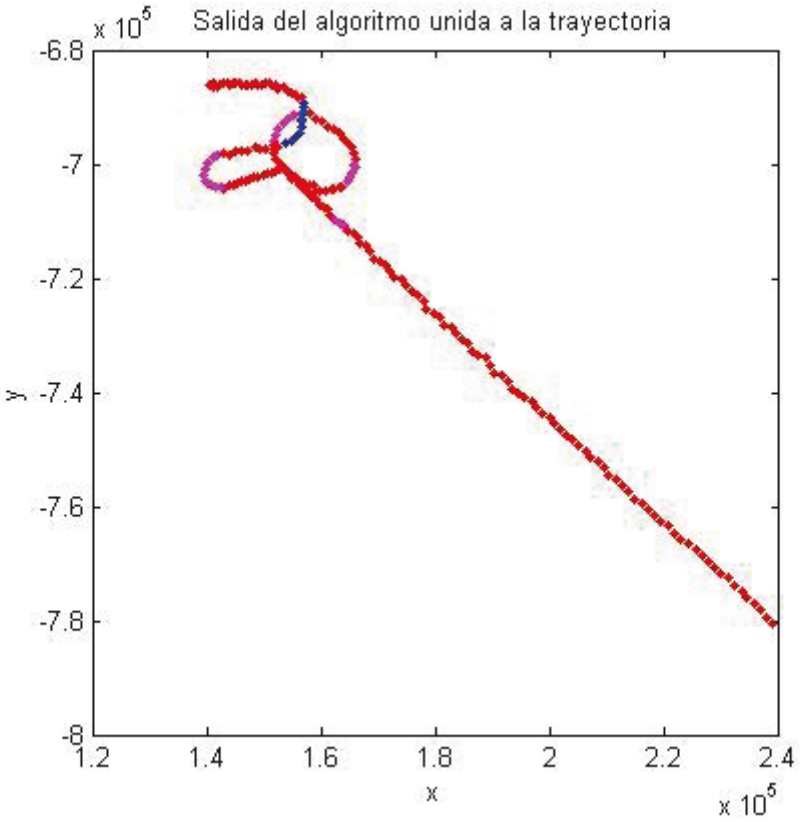


Figura 63: Trayectoria 2 sin filtro de Kalman

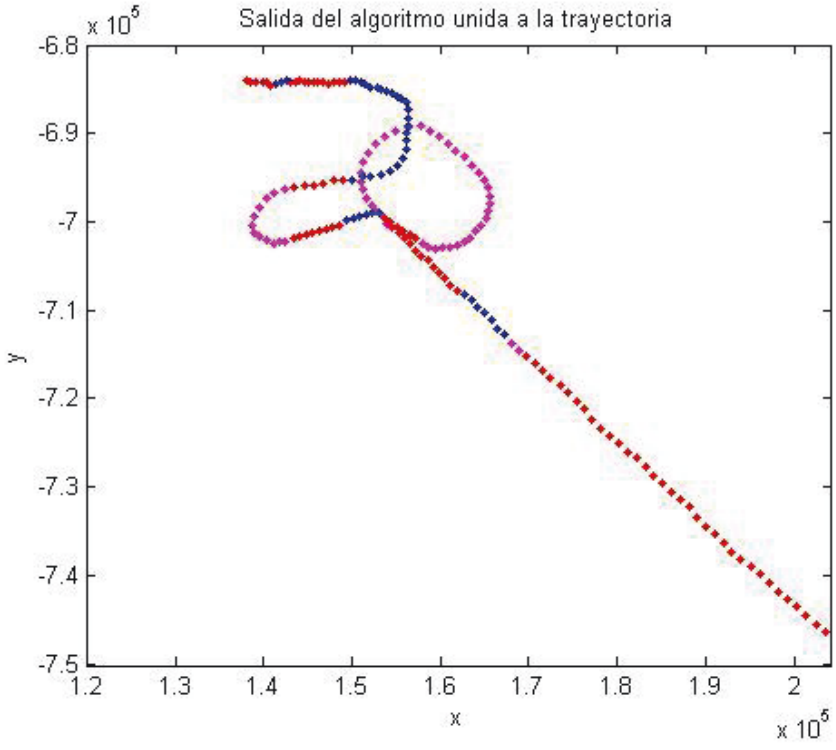


Figura 64: Trayectoria 2 con filtro de Kalman

Tenemos el mismo problema con los hipódromos que con la trayectoria 1. Sin aplicar el filtro, los datos no toman toda la curva, pero al aplicar el filtro toman todo el hipódromo como una curva. Sin embargo, se observa cómo al usar el filtro de Kalman en este caso, el resultado es mejor, ya que consigue distinguir una curva más y ampliar el tamaño de las curvas para que se aproxime más al real.

En las dos ocasiones se comete un pequeño fallo en la clasificación al finalizar la recta más larga, ya que clasifica una pequeña parte como curva.

Trayectoria 3

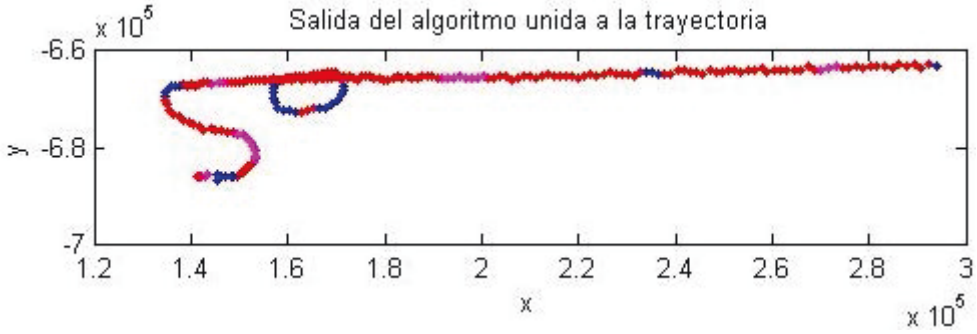


Figura 65: Trayectoria 3 sin filtro de Kalman

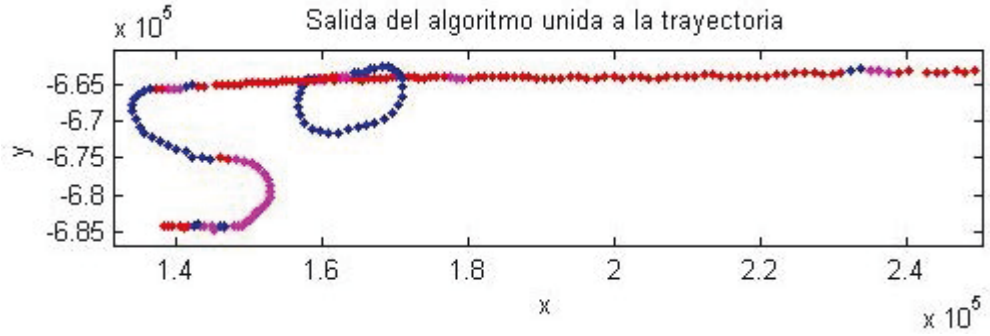


Figura 66: Trayectoria 3 con filtro de Kalman

Seguimos teniendo los mismos problemas con los hipódromos, aunque en este caso realiza mejor la clasificación en la parte del hipódromo sin aplicar el filtro. Pero en las curvas siguientes la cosa cambia, ya que sin aplicar el filtro la clasificación no es tan buena como esperábamos, debido a que el ruido es muy elevado por la distancia al radar, y gracias al filtro la clasificación mejora enormemente.

Trayectoria 4

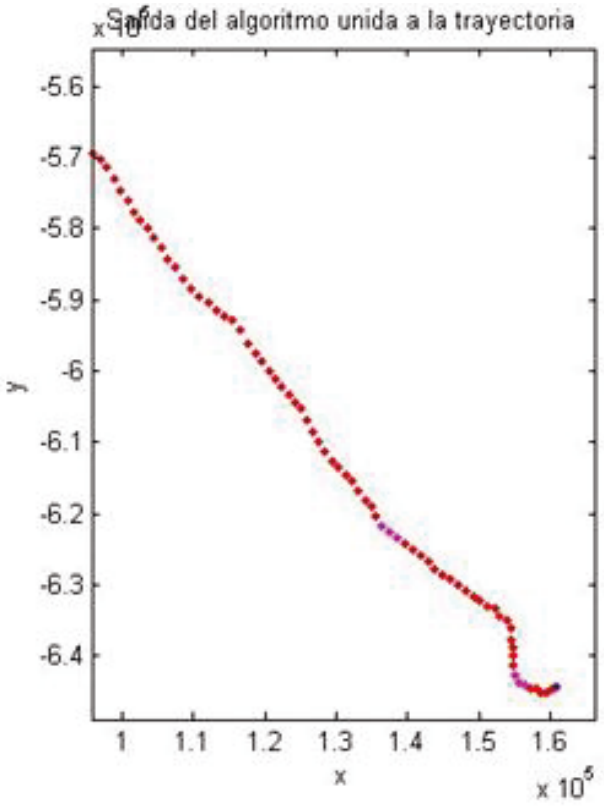


Figura 67: Trayectoria 4 sin filtro de Kalman

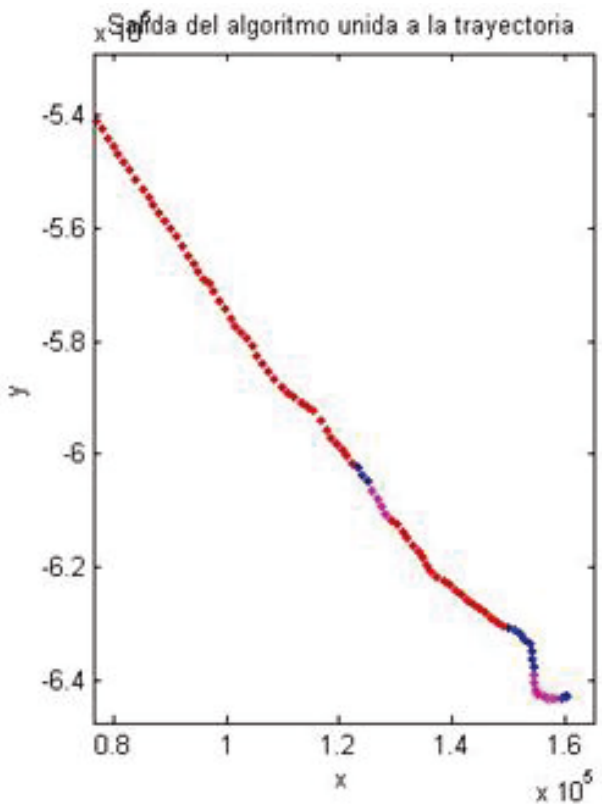


Figura 68: Trayectoria 4 con filtro de Kalman

La clasificación de esta trayectoria es más sencilla cuando se aplica el filtro de Kalman ya que necesitamos suavizar el efecto del ruido, dado que las curvas son más suaves que en las anteriores trayectorias, por lo que funciona mejor cuando el ruido es menor ya que si no complicaría el distinguir las curvas del ruido.

15. Marco regulador y socio-económico

15.1. Restricciones

El proyecto deriva de otro proyecto contratado por la institución Eurocontrol con un consorcio del que formaba parte la universidad Carlos III, específicamente con el GIAA.

Por motivos especificados en el contrato, todas las trayectorias reales no se pueden publicar en coordenadas de longitud y latitud, al igual que la situación real de los radares.

15.2. Presupuesto

El presupuesto necesario para el proyecto es sencillo ya que no requiere grandes inversiones, ni equipos especializados.

Tras una series de estimaciones sobre las horas del proyecto, se estima que la duración del mismo son 1500 horas.

Costes derivados de material

Material	Unidades	Coste / Unidad	Coste
PC´s	1	1.000 €	1.000 €
Material de oficina	-	-	100 €
Total			1.100 €

Tabla 92: Costes de material

Costes derivados de personal

Puesto	Precio/hora	Horas/mes	Horas totales	Coste
Ingeniero colaborador	6 €	110	660	3960 €
Ingeniero colaborador	6 €	110	660	3960 €
Jefe de proyecto	20 €	24	146	2920 €
Total				10840 €

Tabla 93: Costes de personal

Sabiendo que el proyecto tiene una duración aproximada de seis meses después del cálculo de horas y de los costes, tendremos unos costes totales de **11940 €**.

16. Conclusiones y futuras mejoras

Como ya hemos mencionado anteriormente, el objetivo principal de este proyecto era probar la eficacia de los HMM a la hora de clasificar trayectorias reales de vuelo de aeronaves.

- Al analizar el caso base que es un ejemplo académico, hemos visto que todo funcionaba correctamente. Tenía algún problema cuando la serie temporal era muy pequeña, pero obtiene muy buenos resultados, como era de esperar.

- Al clasificar las trayectorias ideales, no hubo ningún problema, quizás a la hora del pre-procesamiento de datos fue algo más costosa, pero una vez conseguimos obtener y discretizar las diferencias angulares, no tuvimos ningún problema.

- Hemos ido avanzando y viendo cómo podíamos enfocar el estudio y qué cambios podíamos realizar para mejorar la eficiencia en la clasificación.
  El primer problema se presentó cuando quisimos entrenar los modelos con la función ***dhmm_em***, ya que sobreentrenaba el modelo debido a la escasez de datos, y si dejaba pocos ciclos de entrenamiento no conseguía buenos resultados. Vimos que su uso no era viable después de varias pruebas, y mediante el método de prueba y error pudimos ajustar manualmente los modelos.

- Después, cuando analizamos los resultados del modelo con los distintos grados de ruido, al observar la entrada del modelo, vimos que no era problema del algoritmo, sino que el ruido afectaba enormemente a las diferencias angulares y hacía imposible una clasificación coherente, por lo que buscamos alguna solución y vimos que si a las trayectorias se le aplicaba previamente el filtro de Kalman reducía excesivamente el efecto del ruido, aunque nos surgió el inconveniente del retraso en las maniobras. No obstante, fue una buena solución para aumentar el porcentaje de aciertos.

- Cuando empezamos a trabajar con los datos reales, vimos que el ruido no era muy elevado, y los resultados eran buenos.
  Pero el principal inconveniente era la diferencia de ruido que nos encontrábamos, teníamos un ruido muy elevado cuanto más se alejaba del radar, y muy poco ruido cuando nos acercábamos. Esto no permitía trabajar bien al algoritmo ya que está cambiando las condiciones en las que se recogen los datos y el algoritmo no podía reconocer las maniobras en esas zonas. Aplicando el filtro de Kalman nuevamente conseguimos una notable mejoría, pero aun así en las zonas donde se encuentra el ruido más elevado, cuando la maniobra es una recta (en la que más perjudica el efecto del ruido) el algoritmo se confunde, y en ocasiones clasifica como giro.

- Como valor añadido, el proceso aquí seguido y los resultados obtenidos han sido utilizados y expuestos para una exposición sobre los Modelos ocultos de Markov y sus aplicaciones en la plataforma OpenCourseWare (OCW) de la UC3M.

Desde mi punto de vista, creo que se conseguiría mejorar los resultados si pudiéramos encontrar algún algoritmo de entrenamiento más efectivo que sea capaz de entrenar los modelos sin necesidad de una cantidad de datos excesiva.

Asimismo, opino que es posible optimizar los resultados con el uso de múltiples HMM para dividir el problema.

Del mismo modo, si además de las diferencias angulares, usáramos otro modelo que clasificara las aceleraciones, se podrían estimar de una manera más favorable las maniobras de las trayectorias.

17. Referencias

[1] Lawrence R. Rabiner. "A tutorial on Hidden Markov Models and selected applications in speech recognition". Proceedings of the IEEE 77, vol 2, pp 257–286, Feb. 1989.

[2] Peña D. "Análisis de series temporales alianza". Alianza, 2005.

[3] Durbin R., Eddy S., Krogh A., Mitchison G. "Biological sequence analysis" pp 166–169, May. 1998.

[4] Juan Besada, E. Voet, J. Garcia, G. Miguel, A. Soto. "ATC trajectory reconstruction for automated evaluation of sensors and trackers performance" . IEEE Aerospace and Electronic Systems Magazine. Vol. 28, N2. pp 4-17, Feb. 2013.

[5] Jensen, Finn V. "An introduction to Bayesian Networks". UCL Press, 1996.

[6] Isasi Viñuela, . "Redes de neuronas artificiales : un enfoque práctico". Pearson Prentice Hall, 2004.

[7] Catlin, Donald E. "Estimation, control, and the discrete Kalman filter". Springer, 1989.

[8] Norris, J.R. "Markov Chains". Cambridge University Press, 1998.

[9] Kevin Murphy. "Hidden Markov Model (HMM) Toolbox for Matlab". <http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html> [Consulta: 15 mayo 2014].

[10] “Hidden Markov Models and the Baum–Welch Algorithm”, IEEE Information Theory Society Newsletter, Dec. 2003.

[11] Proyecto “TÉCNICAS INTELIGENTES APLICADAS A LA EVALUACIÓN DE PROCESADORES DE DATOS PARA EL CONTROL DE TRÁFICO AÉREO”, en el marco del proyecto europeo “TRES: Trajectory Reconstruction and Evaluation Suite“, suscrito con EUROCONTROL (European Organization for Safety of Air Navigation). (Junio 2006- Junio 2008).

[12] Jesús García, Juan A. Besada, Andrés Soto and Gonzalo de Miguel “Opportunity Trajectory Reconstruction Techniques for Evaluation of ATC Systems”. International Journal of Microwave and Wireless Technologies. Volume 1, Special Issue 03, June 2009, pp 231-238 Special issue on surveillance systems for air and airport traffic control In press (2009) ISSN: 1759-0787

[13] Jose Luis Guerrero, Jesus Garcia and Jose Manuel Molina, “Air Traffic Trajectories Segmentation Based on Time-Series Sensor Data" in the book "Sensor Fusion and its Applications", edited by Ciza Thomas. Sciyo 2010. ISBN 978-953-307-101-5

[14] Sidorova J, Aler R, Garcia J. Markov models and hidden markov models. OCW tutorial. UC3M. 2014.

[15] Sidorova, J., Caltabiano, L., Drougov, A., Campillo, M. DUPROSY: dual probabilistic system for biochemical activity prediction. The 8th Int. Conf. On Computing Technology and Information Management. ICCM-2012, vol. 2., pp. 800-803. IEEE. 2012.

[16] Sidorova, J., Anisimova M. NLP-inspired stuctural pattern recognition for a chemical application. Pattern Recognition Letters 45, 11-16, 2014.